8.1. Entorno comando Windows

Los comandos se introducen tecleándolos tras el denominado **indicador del sistema operativo** o el **símbolo del sistema** (llamado ***prompt).***

Cuando ejecutamos un terminal de entorno comando, aparecerá una C u otra letra seguida del símbolo >. La letra dependerá del lugar desde el que se haya realizado la carga del sistema operativo. Si lo hacemos desde el disco duro, aparecerá **C>**; es lo que en la unidad anterior denominábamos unidades lógicas y veíamos a qué unidades físicas estaban asociadas.

Dichos comandos se introducen tras este indicador y para ejecutarlos basta con pulsar la tecla **Enter.**

Los comandos se pueden clasificar en dos grandes bloques, que veremos a continuación. Lo característico de ellos es que, en la mayoría de los casos, tienen que ir acompañados de parámetros o modificadores, produciendo resultados diferentes.

Los comandos que se pueden ejecutar en un terminal Windows se dividen en dos grupos claramente diferenciados. En primer lugar, están aquellas órdenes que son cargadas en memoria al efectuar la carga del sistema operativo y las que no lo son.

Así, las órdenes se dividen en:

- **Órdenes internas.** Son aquellas que han sido cargadas en memoria con la carga del núcleo del propio sistema operativo (ficheros IO.SYS y MS-DOS.SYS). Estas órdenes son las que se incluyen dentro del procesador de comandos o intérprete de comandos (COMMAND.COM).

  Estas instrucciones pueden ser ejecutadas en cualquier momento sin necesidad de realizar ninguna operación extra, ya que están cargadas en la RAM del ordenador.

  Estas órdenes se pueden agrupar en cuatro bloques generales:

  - **Archivos de procesos por lotes.** Son órdenes que sirven para ejecutar procesos **BATCH** en los ficheros **bat.** Algunas de estas órdenes son CALL, GOTO, REM, PAUSE, FOR, etcétera.
  - **Directorios.** Son aquellas órdenes que sirven para manipular directorios, bien para crearlos (MD), borrarlos (RD) o cambiar de directorio (CD).
  - **Ficheros.** Son aquellas órdenes que nos permiten realizar operaciones con y sobre los ficheros. Algunas pueden servir para visualizar el contenido de un archivo (TYPE), para eliminarlo (DEL, DELETE, REMOVE) y para ver la estructura de archivos y ficheros (DIR).
  - **Otras.** Son aquellas órdenes que sirven para establecer el entorno de trabajo del usuario. Pueden indicar las trayectorias o caminos en los que se bucarán determinados archivos (PATH) o modificar el aspecto del indicador del sistema (PROMPT), entre otras.

- **Órdenes externas.** Son aquellas que no se han cargado en RAM. Físicamente tienen que residir en el disco y para ser ejecutadas el ordenador tiene que realizar la operación de lectura correspondiente del disco. Es por eso por lo que la ejecución de estas órdenes es más lenta que en el caso de las órdenes internas, ya que es necesario la carga de la orden en memoria para que ésta pueda ser ejecutada.

  Estas órdenes tienen un nombre de hasta ocho caracteres y siempre tendrán la misma extensión: **COM** o **EXE**.

Independientemente de que la orden sea interna o externa, siempre que necesitemos conocer la sintaxis y los parámetros de la misma podemos solicitar ayuda al sistema operativo.

Una forma de solicitar ayuda sobre una orden concreta consiste en teclear el nombre de la orden después del símbolo del sistema y, a continuación, **/?** De esta forma, se nos mostrará por pantalla toda la sintaxis de la instrucción. No todas las órdenes o comandos pueden mostrar la información de ayuda con esta opción.

Otra forma de solicitar ayuda de las órdenes del sistema operativo de forma conjunta es utilizar el comando **HELP.** Gracias a él, el usuario podrá buscar la orden concreta para analizar su sintaxis y el uso que tiene en Windows. Con la orden HELP podemos mostrar información completa sobre todos los comandos y órdenes.

Cuando la acción que se va a realizar con un comando se refiere a un directorio que no es el activo, indicaremos, seguida de un espacio en blanco, la unidad lógica y la trayectoria, relativa o absoluta, donde queremos que el comando realice la acción deseada.

Por último, generalmente se incluyen, si es necesario o así lo deseamos, los parámetros para modificar el resultado estándar que la orden realizaría sin ningún modificador.

A la hora de presentar la sintaxis de los comandos, vamos a emplear la misma notación empleada en los manuales de SO. En esta notación, los corchetes ([ ]) indican que aquello que contienen es opcional, es decir, puede aparecer o no, y el pipe (|) indica que es preciso elegir entre el contenido de su derecha o de su izquierda, no pudiendo utilizarse ambos simultáneamente.

La sintaxis general de un comando del SISTEMA OPERATIVO responde a la siguiente estructura:

Comando [unidad lógica] [/parámetro1] [/parámetro2]

8.2. Organización de la información.

Ya se ha comentado que la información se organiza dentro del disco de forma jerárquica o en árbol. Para ello se utilizan los directorios, los cuales, a su vez, pueden contener subdirectorios y ficheros o archivos. Un ejemplo muy claro se indica a continuación:

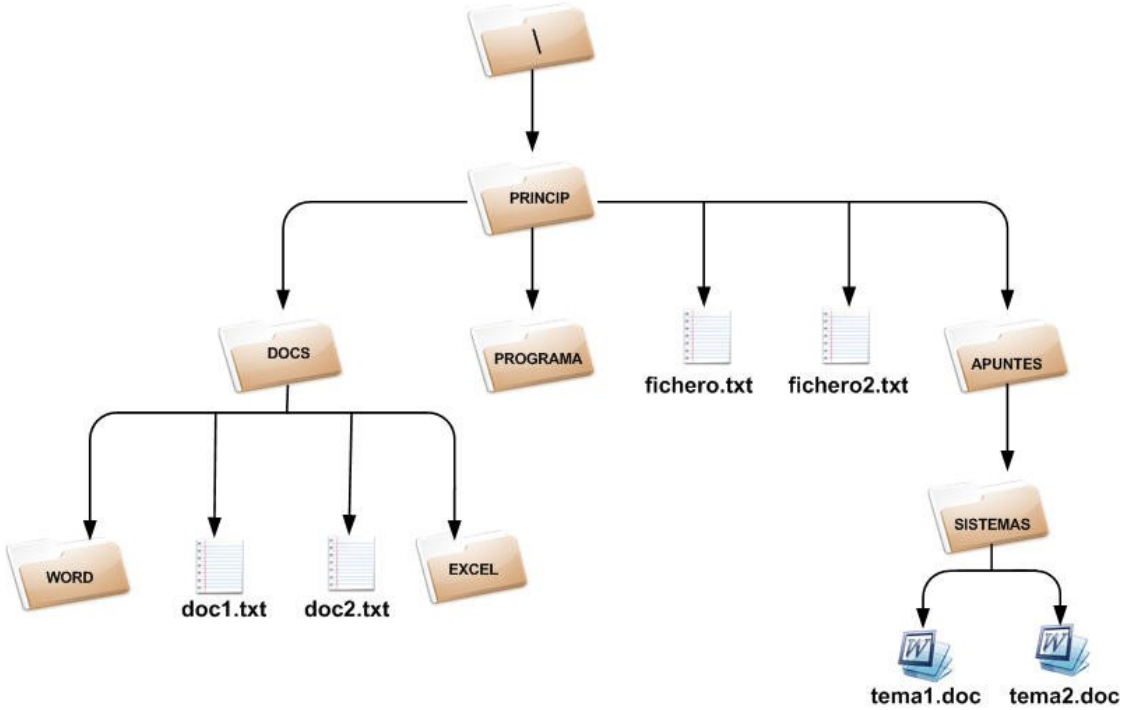


Fig. 1. *Árbol de directorios y ficheros.*

En este caso, la información que tenemos parte siempre del directorio raíz, independientemente de que ésta esté en un disco duro o en un disquete, incluso en un CD-ROM.

Como se puede apreciar en la Figura 1, la estructura es arborescente y jerárquica. Todo cuelga del directorio raíz. Así, PRINCIP es un subdirectorio de raíz. DOCS, PROGRAM y APUNTES son subdirectorios que cuelgan del directorio PRINCIP que, a su vez, cuelga del directorio raíz.

Podemos observar que el directorio PRINCIP, que cuelga del directorio raíz, contiene los subdirectorios anteriormente mencionados además de dos archivos o ficheros: fichero.txt y fichero2.txt.

8.3. Unidad activa y directorio activo.

Existen distintas formas de hacer referencia a un fichero o directorio, dependiendo de que tengamos en cuenta o no en qué parte de la estructura jerárquica estamos situados.

Para empezar a explicar cómo se accede a los ficheros y directorios, hemos de tener en cuenta dos conceptos:

- **Unidad activa.**
- **Directorio activo.**

Se llama **unidad activa** a la letra que indica la unidad lógica con la que se corresponde una unidad física sobre la que el sistema operativo tiene el control o está situado.

Cómo comentamos anteriormente, A: se corresponde con la unidad de disquete, y C: con al unidad de disco duro. Para poder acceder a un fichero o directorio concreto hay que saber en qué unidad estamos situados; es decir, cuál es la unidad activa.

Además, para poder movernos por la estructura de directorios y subdirectorios de la misma unidad o de otra, es necesario saber en qué subdirectorio nos encontramos; es decir, necesitamos conocer el **directorio activo.**

Esta información es necesaria para que el acceso a archivos sea eficaz. Pero, ¿dónde encontrarla? Pues bien, es el propio sistema operativo el que indica cuál es el directorio activo gracias al denominado **símbolo del sistema.** Ya veremos cómo se configura el sistema operativo para que el símbolo del sistema muestre tal información.

Lógicamente, según el usuario cambie de un directorio a otro, o de una unidad a otra, el directorio activo y la unidad activa cambiarán. En cualquier caso, estemos donde estemos, siempre existirá unidad activa y directorio activo.

8.4. Trayectorias o caminos en entorno comando Windows.

A. Definición de trayectoria.

Partiendo de una unidad activa y de un directorio activo, la posición de un fichero o un directorio en nuestro disco se denomina **trayectoria.**

Una trayectoria es un nombre que nos indica la posición de directorios y ficheros. Por otro lado, las trayectorias no afectan a las unidades lógicas. La unidad lógica será A, B, C, etc., seguida de dos puntos (:).

Dentro de las trayectorias debemos tener en cuenta el carácter **\.** Este símbolo sirve para separar los nombres de directorios y subdirectorios.

B. Tipos de trayectorias.

Para explicar los tipos de trayectorias, vamos a partir de que la unidad activa es C: y el directorio activo es DOCS. Las trayectorias en entorno comando pueden ser de dos tipos:

I) Trayectorias absolutas. Con este tipo de trayectorias se puede identificar ficheros y directorios sin tener en cuenta ni la unidad ni el directorio activo.

Partamos del fichero **doc1.txt**. Supongamos que la unidad activa es la referente al disco duro, es decir, **C:**. La trayectoria en la que se encuentra este archivo es: **C:\princip\docs\doc1.txt**

Vemos que una trayectoria absoluta se construye indicando en primer lugar la unidad, seguida de \, para seguir con una cadena de subdirectorios, cada uno de ellos separados por \, hasta llegar a indicar exactamente dónde se encuentra el archivo.

Si la unidad fuese la de disquete y quisiéramos referenciar el director raíz, bastaría con poner: **A:**

Supongamos que queramos indicar la trayectoria en la que se encuentra el fichero **tema1.doc.**

La trayectoria absoluta sería: **C:\princip\apuntes\sistemas\tema1.doc**

II) Trayectorias relativas. Para utilizar estas trayectorias es importante tener en cuenta tanto el directorio activo como la unidad activa.

Es lógico pensar, al contrario del caso de la trayectoria absoluta que será única, que las trayectorias relativas serán múltiples. Dependerá del directorio y unidad activa en la que nos encontremos que la trayectoria sea diferente.

Antes de poner algún ejemplo de estas trayectorias, tenemos que hablar de dos archivos importantes que contienen siempre todos los subdirectorios, a excepción del directorio raíz.

Cada directorio en Windows, una vez que es creado, está vacío; es decir, no tiene ni subdirectorios ni archivos o ficheros que cuelguen de él. Pero, al crear un directorio, Windows, automáticamente, genera dentro de él dos subdirectorios. Son los siguientes:

. (punto). Este subdirectorio hace referencia al contenido del propio directorio.

.. (punto, punto). Este subdirectorio hace referencia al **directorio padre.** El directorio padre es aquel que se encuentra jerárquicamente por encima del directorio en el que estamos situados. Es por eso por lo que el directorio raíz no contendrá este fichero, ya que por encima de él no hay ningún otro directorio en la jerarquía.

Una trayectoria es la que se construye partiendo desde donde estamos, bajando por la estructura jerárquica para llegar al directorio o fichero que deseamos.

Ejemplo.

Supongamos que estamos en el directorio DOCS y queremos indicar la trayectoria en la que se encuentra el fichero tema1.doc. Habría que teclear:

..\apuntes\sistemas\tema1.doc

A continuación, y partiendo desde SISTEMAS, vamos a indicar la ruta o trayectoria en la que se encuentra el directorio EXCEL. Habría que teclear:

..\..\docs\excel

Como se puede ver, hemos ascendido por el árbol hasta PRINCIP para, a continuación, bajar hasta EXCEL.

Cuando veamos las órdenes de manejo de directorios, y en especial la orden CD, insistiremos más en esta cuestión.

8.5. Órdenes generales en entorno comando de Windows

http://commandwindows.com/command1.htm

En este punto hablaremos de órdenes del sistema operativo que no se utilizan específicamente para el manejo de ficheros y/o directorios. Son órdenes para gestionar la interfaz de texto, para realizar modificaciones en parámetros del sistema, etcétera.

ORDEN/ COMANDO	SINTAXIS	ACCIÓN
CLS (interno)	**CLS**	Este comando limpia la pantalla de la información que muestra en ese momento. Tras ejecutarlo, solamente quedará en la primera fila de la pantalla el indicador del sistema indicando la unidad activa y directorio activo
DATE (interno)	**DATE [FECHA (según formato)]**	Es el comando que visualiza la fecha actual del sistema y permite modificarla
TIME (interno)	**TIME [HORAS : MINUTOS [SEGUNDOS [CENTÉSIMAS]]]**	Es el comando que muestra la hora actual del sistema y permite modificarla.
HELP (externo)	**HELP [COMANDO]**	Muestra una descripción o ayuda indicando la sintaxis y parámetros necesarios para poder usar cualquier orden de sistema operativo.
VER (interno)	**VER**	Es el comando que visualiza el número de versión del sistema operativo que se está utilizando.
PROMPT (interno)	**PROMPT [TEXTO] [$MODIFICADORES]**	Indica cuál es el símbolo del sistema. Al encender el equipo y cargar el sistema operativo, la pantalla del ordenador únicamente muestra dos cosas: • La unidad lógica desde la que se ha realizado la carga del sistema operativo. • El carácter >. Con el comando **prompt** se puede modificar el símbolo del sistema por defecto. El comando incluye varios parámetros propios aunque podemos indicar cualquier cosa, incluso un texto apareciendo este como símbolo del sistema. Si utilizamos los modificadores o parámetros propios del comando, tendremos que ponerlos precedidos cada uno de ellos con símbolo **$.** **MODIFICADORES** **t** hora actual **d** fecha actual **v** versión del

		sistema operativo. **p** camino y unidad actual **n** nombre de unidad actual **l** carácter < **g** carácter > **q** carácter = **b** carácter |

8.6. Órdenes de manejo de discos en entorno comando.

http://commandwindows.com/command1.htm

A. Particiones de disco.

Particionar un disco duro es una de las operaciones que previamente hay que realizar para instalar cualquier sistema operativo.

En primer lugar, para poder instalar cualquier sistema operativo Windows, UNIX, LINUX o similar, es necesario particionar el disco duro, ya que de fabrica solamente trae la estructura física. El particionado consiste en una operación en la que se indica qué espacio del disco duro como máximo y mínimo vamos a destinar del dispositivo para instalar el sistema operativo, entre otras cosas.

Las operaciones de particionado en sistemas Microsoft DOS o Windows, se realizaban con la orden **FDISK.EXE.**

Actualmente, en entorno comando en Windows XP o Windows 7 se utiliza la orden **diskpart.**

Las particiones se realizan para indicar al sistema operativo cuál será el tamaño que deseamos utilizar del disco. Podemos utilizar todo o **parte** del él. También se realizan particiones para definir las diferentes unidades lógicas en otra parte del disco, si así lo deseamos. Esta orden solamente se puede utilizar en discos duros y no en disquetes, CD-ROM, cintas, etcétera.

Hay que tener al menos, una partición en nuestro disco. Esta partición obligatoria, denominada **partición primaria,** siempre tiene que existir. Puede ser que la definamos sin utilizar todo el espacio del disco duro. Esto ocurrirá si, en un momento determinado, queremos realizar una segunda partición, la **partición extendida,** para instalar otro sistema operativo diferente o para cualquier otra función.

Antes de dar formato lógico al disco; es decir, antes de establecer la estructura de *boot,* FAT, zona de directorios y zona de datos, es obligatorio particionarlo. Las particiones las puede realizar el propio usuario o dejar al sistema operativo para que sea el proceso de instalación quien se encargue de realizar esta labor. Las particiones solamente se pueden realizar sobre los discos duros y no sobre los disquetes.

Una partición es un conjunto de caras, pistas y sectores (en su caso, cilindros) independientes, que forman una unidad física, también independiente, a la que estará asociada una unidad lógica, según el sistema operativo que instalemos.

Las particiones tienen varias funciones. Una de ellas es poder instalar en un mismo disco duro varios sistemas operativos. Si cada sistema operativo lo instalamos en una partición diferente, cada uno de ellos será independiente. Otra utilidad es la de tener varias unidades lógicas dentro del mismo disco duro; tener varias unidades lógicas con diferentes sistemas de archivos, etc. De esta forma, el usuario podrá utilizar su máquina con diferente software básico o bien entender que tiene más de un disco duro. Además, permite dividir la unidad física destinada a un mismo sistema operativo en varias unidades lógicas.

La forma de proceder para preparar el espacio de almacenamiento de un disco duro es la siguiente: en primer lugar habrá que hacer las correspondientes particiones utilizando el

comando **FDISK** (o el equivalente en cada sistema), que incorpora el sistema operativo en el primer disquete. Realizadas las particiones correspondientes, procederemos a formatearlas con el comando **FORMAT** o con el comando correspondiente de cada sistema operativo.

Siempre que se realizan particiones, una de ellas ha de tener la característica de ser **una partición activa.** Esta partición será la encargada de tomar el control del sistema cuando se inicializa el ordenador.

En realidad existen dos tipos de particiones:

- **Partición primaria.**
- **Partición extendida.**

La **partición primaria** es obligatoria y en ella se encuentran (aunque no necesariamente) los ficheros de arranque del sistema. Esta será la partición activa, siempre y cuando no exista otro sistema operativo almacenado en el mismo disco duro. Esta partición primaria se corresponderá con una unidad lógica que, normalmente, será C:.

La partición primaria tendrá un tamaño que va a depender de dos aspectos. En primer lugar, de la versión del sistema operativo que estamos utilizando y, en segundo, del sistema de archivos con el que queramos utilizar la partición.

La **partición extendida** sirve para dividir el espacio del disco duro en varias unidades lógicas. Esta partición puede dividirse en una o más unidades lógicas que no son particiones, ya que no pueden almacenar distintos sistemas operativos, aunque son reconocidas por el sistema operativo como diferentes unidades lógicas.

B. Dar formato a un disco.

FORMAT *(orden externa)*	
FORMAT volumen [/FS:sist_arch] [/V:etiqueta] [/Q] [/A:tamaño] [/C] [/X] [/P:ciclos] [/S:estado] FORMAT volumen [/V:etiqueta] [/Q] [/F:tamaño] [/P:ciclos] FORMAT volumen [/V:etiqueta] [/Q] [/T:pistas /N:sectores] [/P:ciclos] FORMAT volumen [/V:etiqueta] [/Q] [/P:ciclos] FORMAT volumen [/Q]	
PARÁMETROS	
volumen	Especifica la letra de unidad (seguida por dos puntos), el punto de montaje o el nombre de volumen.
/ FS:sistemaDeArchivos	Especifica el tipo de sistema de archivos (FAT, FAT32, exFAT, NTFS o UDF).
/Q	Realiza un formato rápido de disco. Borra la FAT y el directorio raíz de un disco que tuvo formato previo; no busca sectores defectuosos. Este modificador invalida /P.
/V: etiqueta	Especifica la etiqueta del volumen.
/S	Copia en el disco que va a ser formateado los ficheros del MS-

	DOS: IO.SYS, MS-DOS.SYS Y COMMAND.COM. Hace que el disco sea de arranque de MS-DOS.
/autotest	Formato incondicional
/C	Solo NTFS. los archivos creados en el nuevo volumen se comprimen predeterminadamente.
/X	Si es necesario, obliga al volumen a desmontarse primero. Todos los controladores abiertos al volumen serán no validos.
/R: revisión	Solo UDF: fuerza a que se formatee con una versi¢n UDF específica (1.02, 1.50, 2.00, 2.01, 2.50). La revisión predeterminada es 2.01.
/D	Solo UDF 2.50: se duplicarán los metadatos.
/A: tamaño	Invalida la asignación predeterminada del tamaño de unidad. Se recomienda usar la configuración predeterminada para uso general: • NTFS admite 512, 1024, 2048, 4096, 8192, 16.000, 32.000, 64.000. • FAT admite 512, 1024, 2048, 4096, 8192, 16.000, 32.000, 64.000 (128.000, 256.000 para tamaño de sector > 512 bytes). • FAT32 admite 512, 1024, 2048, 4096, 8192, 16.000, 32.000, 64.000 (128.000, 256.000 para tamaño de sector > 512 bytes). • exFAT admite 512, 1024, 2048, 4096, 8192, 16.000, 32.000, 64.000, 128.000, 256.000, 512.000, 1.000.000, 2.000.000, 4.000.000, 8.000.000, 16.000.000, 32.000.000. Tenga en cuenta que los sistemas de archivos FAT y FAT32 imponen las siguientes restricciones en el número de *clústeres* en un volumen: • FAT: Número de *clústeres* <= 65526 • FAT32: 65526 < Número de *clústeres* < 4177918 Se detendrá inmediatamente el proceso de formato si decide que los requisitos anteriores no se cumplen con el tamaño de *clúster* especificado. No se admite la compresión de NTFS para asignar tamaños de unidades superiores a 4 096.
/F: tamaño	Indica el tamaño del disquete al que se da formato. Esta capacidad viene expresada en KB, pudiendo ser 160, 180, 320, 360, 720, 1 200, 1 400 o 2 880.
/T: pistas	Especifica el número de pistas por cara del disco.
/N: sectores	Especifica el número de sectores por pista.
/P: ciclos	Llena con ceros cada sector de los ciclos realizados en el volumen. Este modificador no es válido con /Q.

/S: estado	Donde “estado” es “enable” o “disable”. De forma predeterminada, los nombres cortos están habilitados.

Ejemplo.

C:\FORMAT a:

Formatea un disquete estándar en la unidad A:, según el tamaño del soporte y características de la disquetera.

Si queremos formatear un disquete para que tenga cargado el núcleo del sistema operativo, teniendo en cuenta que es un sistema disco de 720 Mb, el resultado sería el siguiente:

C:\>FORMAT a: /S /F:720

Es importante tener en cuenta que el parámetro **/q** solamente se puede utilizar cuando el disco ha sido formateado al menos una vez previamente. Cuando compramos un disco nuevo y no está formateado, este parámetro no puede utilizarse, ya que el disco aún no tiene creada una estructura lógica de ningún tipo.

C. Otras ordenes de discos.

SYS *(orden externa)*

SINTAXIS

SYS [UNIDAD1:] [RUTA] UNIDAD2:

Transfiere los ficheros fundamentales del sistema operativo a la unidad especificada, de forma que se crea un disco con el cual podemos inicializar nuestro ordenador. Recordemos que estos ficheros son el IO.SYS, MSDOS.SYS y el intérprete de comandos COMMAND.COM.

Los discos tienen todos la mismas función: almacenar datos, pero hay discos que, además de esa utilidad, pueden tener otra: ser discos de sistema. Estos discos pueden ser disquetes o **discos duros,** incluso unidades de CD-ROM. La condición necesaria es que tengan los tres ficheros antes mencionados en el directorio raíz.

Los ficheros IO.SYS y MSDOS.SYS son ficheros ocultos. Siempre que formateamos un disco con el parámetro /S o cuando instalamos el sistema operativo, estos ficheros se introducen el directorio raíz, además del COMMAND.COM.

Si queremos hacer que un disco sea de sistema, tendremos que utilizar el comando SYS.

Ejemplo.

C:\>SYS a:

De esta forma hacemos que el disquete insertado en la unidad A: sea un disco de sistema, ya que transferimos los archivos necesarios.

Los archivos de sistema (IO.SYS y MSDOS.SYS) deben estar en el directorio raíz o en el directorio desde el que se ejecute la orden SYS; por eso, no especificamos necesariamente

la unidad de origen, ya que estamos situados en el lugar adecuado. Si estuviéramos en otro sitio, tendríamos que poner lo siguiente:

C:\princip>SYS c:\ a:

Indicando que los archivos de sistema están en el sitio que les corresponde y que es diferente al que estamos situados.

Ejemplo.

Vamos a crear un disco con núcleo de sistema operativo; es decir, vamos a crear un disco de arranque.

La operación la podemos realizar en entorno comando de dos formas, con el mismo resultado:

a) **C:\>FORMAT a: /S**

b) **C:\>FORMAT a:**

C:\>SYS a:

La misma operación es válida para discos duros; es decir, si tenemos un disquete con los archivos de sistema y tenemos el comando SYS, podemos transferir el sistema al disco duro poniendo:

A:\>SYS c:

Ya veremos, en el proceso de inicialización del sistema, cómo se realiza la carga del núcleo y de otros archivos. Además veremos desde qué unidad se realiza tal proceso.

DISKCOPY (externa)

SINTAXIS

DISKCOPY [unidad1: [unidad2:]] [/V]

- Los dos disquetes deben ser del mismo tipo.
- Puede especificar la misma unidad para unidad1 y unidad2.

/V	Comprueba que la información se copie de forma correcta.

Con este comando podemos duplicar un disquete en otro, siempre y cuando sean de la misma capacidad. No se puede utilizar con discos duros. Si quisiéramos duplicar un disco duro en otro, solo podremos hacerlo utilizando la orden **XCOPY** para duplicar toda la estructura de un disco duro.

La copia de discos es como una fotocopia. Se copia exactamente lo mismo de un disco a otro, incluso lo que pueda estar mal en el disco original.

Si disponemos en Windows de dos disqueteras o unidades de disquete, la copia la podremos hacer en una sola operación; es decir, introduciendo el disco origen en una unidad y el destino de origen en una unidad y el destino en otra.

Para copiar un disco en otro con unidades de disquete, la orden sería la siguiente:

C:\>DISKOPY a: b:

Ahora bien, si solo tenemos una unidad de disquete, la copia se hace un poco más lenta, ya que tenemos que utilizar la misma unidad para insertar el disco de origen y el de destino.

C:\>DISKCOPY a: a:

Cuando ponemos la orden de esta forma, la copia es igual de efectiva, ya que se nos pedirá en primer lugar que introduzcamos el disco de origen en la unidad A:. Una vez leído todo el disco (en versiones superiores a la 5.0), la información queda temporalmente almacenada en el disco duro. Tras esto, se nos pedirá que introduzcamos el disco de destino en la misma unidad. La información almacenada temporalmente en el disco duro pasará al nuevo disco.

Es importante tener en cuenta que, si el disco de destino no está formateado previamente, la orden procederá a darle formato antes de realizar la copia.

LABEL (externa))
SINTAXIS LABEL [UNIDAD:] [ETIQUETA]

Es un comando que crea, cambia o borra la etiqueta de identificación del volumen del disco.

Windows, como la mayoría de los sistemas operativos, identifica las unidades de almacenamiento mediante una etiqueta. Esta etiqueta puede ser un número o un nombre con una extensión no superior a once caracteres, sin espacios en blanco. Así, cuando referenciamos determinada unidad lógica, sabremos cuál es por el nombre que tiene.

Si solo se escribe **LABEL,** el sistema operativo muestra la etiqueta (si la hay) del disco de la unidad activa y solicita que se introduzca una nueva o que pulse **Enter.** Si se hace esto último y el disco tiene ya etiqueta, el sistema operativo pregunta si se desea borrarla.

Ejemplo.
C:\>LABEL El volumen de la unidad C: es MS_DOS_6.2. El número de serie del volumen es 26762590. Etiqueta de volumen (máximo once caracteres, ENTRAR para ninguna).

En este caso, si pulsamos **Enter** damos opción a eliminar la anterior, mientras que si introducimos una nueva, será la etiqueta vigente para este volumen o unidad de disco.

C:\>LABEL a: programas

A la unidad A: le pone la palabra **programas** como etiqueta identificativa.

VOL (externa)
SINTAXIS

VOL [UNIDAD:]

Permite solo visualizar la etiqueta del disco. La función de este comando la realiza también el comando **LABEL.**

Ejemplo.
C:\>VOL A: **El volumen de la unidad A: es PROGRAMAS.** **El número de serie del volumen es 24362752.**

8.7. Órdenes de manejo de directorios en entorno comando.

http://commandwindows.com/command1.htm

A. Introducción a los directorios.

Ya sabemos que un directorio es un fichero que puede contener otros directorios (subdirectorios) o archivos regulares.

Con los directorios podemos almacenar información de forma discriminada. Tendremos directorios para contener documentos, otros para contener programas, otros para guardar imágenes gráficas, etc. Es una forma de estructurar la información almacenada en nuestro soporte de almacenamiento externo.

Cada directorio aparece dentro de la estructura con un nombre que no puede ser superior a 8 caracteres y que puede contener una extensión de tres caracteres separados del nombre por un punto.

Los nombres de directorio siguen las mismas reglas de formación que los nombres de fichero, pero no suelen tener extensión, aunque pueden tenerla. Con los ficheros ocurre lo mismo, pueden tener un nombre con extensión o sin ella, aunque los archivos o ficheros regulares suelen tener extensión.

Los nombres de directorio, al igual que los de fichero, pueden contener los caracteres de la A a la Z y los dígitos del 0 al 9, sin espacios en blanco ni caracteres especiales. Pueden incluir el guion bajo (_) o el carácter ~ (ASCII 126).

B. Ordenes de manejo de directorios.

DIR (interno) Muestra la lista de subdirectorios y archivos de un directorio.
SINTAXIS DIR [unidad:][ruta][archivo] [/A[[:]atributos]] [/B] [/C] [/D] [/L] [/N][/O[:]orden]] [/P] [/Q] [/S] [/T[[:]fecha]] [/W] [/X] [/4]
/A Muestra los archivos con los atributos especificados.

atributos

- D Directorios.
- R Archivos de solo lectura.
- H Archivos ocultos.
- A Archivos para archivar.
- S Archivos de sistema.
- - Prefijo que significa no.

/B Usa el formato simple (sin encabezados ni sumarios).

/C Muestra el separador de miles en el tamaño de los archivos. Esto es lo predeterminado. Use /-C para deshabilitar la aparición de dicho separador.

/D Como el listado ancho pero los archivos aparecen clasificados por columnas.

/L Usa letras minúsculas.

/N Nuevo formato de lista larga donde los archivos aparecen en el lado derecho.

/O Lista los archivos según lo indicado en orden.

Orden

- N Por nombre (orden alfabético).
- E Por extensión (orden alfabético).
- S Por tamaño (orden creciente).
- D Por fecha y hora (el más antiguo primero).
- G Agrupar primero los directorios.
- - Prefijo para invertir el orden.

/P Hace una pausa después de cada pantalla completa de información.

/Q Muestra el propietario del archivo.

/S Muestra los archivos del directorio especificado y todos sus subdirectorios.

/T Controla el campo de fecha que se mostrará o usará para realizar la clasificación.

fecha

- C Creación.
- A Último acceso.
- W Última modificación.

/W Usa el formato de listado ancho.

/X Muestra los nombres cortos generados para los nombres de archivo que no tienen el formato 8.3. El formato es el de /N con el nombre corto especificado antes del nombre largo. Si no existe un nombre corto, se muestran espacios en blanco en su lugar.

/4 Muestra el año con 4 dígitos

Los modificadores pueden estar prestablecidos en la variable de entorno

DIRCMD. Éstos se pueden anular precediendo cada modificador con - (guion), por ejemplo: /-W.

Lista el contenido de un directorio, mostrando el nombre y extensión de cada archivo, el tamaño en bytes, la fecha y hora en que se creó o modificó por última vez y los subdirectorios que cuelgan de él. Indica también al principio de la lista y como información complementaria, la etiqueta y número de serie y de disco, y al final indica el número y

tamaño del conjunto de archivos que se han mostrado y el espacio que queda libre en el disco.

Con este comando también podemos apreciar que en la lista de directorios aparecerán siempre los directorios **.** y .., excepto en el directorio raíz.

Ejemplo.

Si ejecutamos:

C:\>DIR /W

la información se mostrará en forma de lista. Los nombres que correspondan a directorios no aparecerán entre los símbolos < >, sino entre [ ]. Es evidente que se muestra menos información particular de cada archivo. Por el contrario, en la pantalla aparece más información sobre el número de ficheros y directorios que tenemos en ese momento.

Con la orden:

C:\>DIR /O:GN

se muestra por pantalla un listado, ordenado alfabéticamente, del contenido del directorio raíz agrupando por separado directorios y ficheros.

CD (interno)

Muestra el nombre del directorio actual o cambia de directorio.

SINTAXIS

CHDIR [/D] [unidad:][ruta]

CHDIR [..]

CD [/D] [unidad:][ruta]

CD [..]

.. Especifica que desea cambiar al directorio superior.

Escriba **CD** unidad: para ver el directorio actual de la unidad especificada.

Escriba **CD** sin parámetros para ver la unidad y el directorio actual.

Use el modificador **/D** para cambiar la unidad actual además del directorio actual para una unidad de disco.

Si las extensiones de comando están habilitadas, **CHDIR** cambia así:

El uso de mayúsculas y minúsculas de la cadena del directorio actual se convierte al mismo uso que se tiene en los nombres de unidades. Así, **CD C:\TEMP** establecerá C:\Temp como el directorio actual si éste es el uso de mayúsculas y minúsculas en la unidad.

El comando **CHDIR** no trata los espacios como separadores, así que es posible usar **CD** para cambiar a un directorio cuyo nombre de subdirectorio contenga un espacio, sin necesidad de escribir el nombre entre comillas. Por ejemplo:

cd\winnt\profiles\nombredeusuario\programas\menú inicio

es lo mismo que:

cd "\winnt\profiles\nombredeusuario\programas\menú inicio"

que sería lo que hay que escribir si las extensiones estuvieran deshabilitadas.

Con esta orden podemos movernos por la estructura de directorios que tenemos creada. Sirve para subir o bajar por la estructura jerárquica.

Esta orden puede incluir o no parámetros.

Si ejecutamos la orden **CD** sin parámetros, nos muestra el directorio y unidad activa en la que estamos situados. Esta orden sin parámetros no suele utilizarse, ya que se supone que el **prompt** del sistema estará indicando directorio y unidad activa siempre.

Ejemplo.

Supongamos que no tenemos activado el prompt para indicar directorio y unidad activa. Supongamos que estamos situados en el directorio PROGRAM de nuestra estructura. El resultado de ejecutar la orden **CD** sin parámetros es el siguiente:

C>CD

C>C:\princip\program

Lo normal es tener activado el prompt de tal forma que apareciese lo siguiente:

C:\princip\program>

De esta forma sabremos siempre dónde estamos situados.

EJEMPLO

Si ejecutamos la orden:

C:\princip\program>CD A:

Y nos aparece:

A:\basic>

Esto indica que en la unidad de disquete, estamos situados en un directorio llamado BASIC.

Si indicamos **CD** seguido de un nombre de directorio, indicaremos que queremos cambiar a ese directorio. Para ello tendremos que tener en cuenta que el directorio al que queremos cambiar dependa del directorio y unidad en la que estamos situados.

En otro caso, tendremos que indicar la trayectoria relativa o absoluta para llegar al directorio deseado.

Si ejecutamos la orden **CD** seguida de una letra correspondiente a una unidad lógica, se visualiza la unidad la unidad que estamos referenciando y el directorio activo de ésta.

Vamos a seguir practicando la orden **CD** y sus variantes.

Ejemplo.

Supongamos que estamos situados en el directorio PRINCIP de nuestro ejemplo:

C:\princip>

Y queremos cambiarnos al directorio PROGRAM. Como éste está por debajo del directorio PRINCIP, basta con que pongamos lo siguiente:

C:\princip>CD program

De esta forma, nos situamos en el directorio deseado:

C:\princip\program>

En este caso, el directorio PRINCIP se denomina directorio padre y el directorio PROGRAM, directorio hijo. En este mismo caso, el directorio raíz sería el directorio padre de PRINCIP y éste, el hijo del directorio raíz.

Pero supongamos que estamos situados en el directorio raíz y queremos situarnos en el directorio PROGRAM. Las formas de proceder son las siguientes:

1. Situarnos primero en PRINCIP, y luego en PROGRAM:

C:\>CD princip

C:\princip>CD program

C:\princip\program>

2. Indicar la trayectoria completa, absoluta o relativa, desde el directorio raíz, para llegar al directorio deseado:

C:\>CD princip\program

C:\princip\program>

O bien:

C:\>CD c:\princip\program

Igual que nos situamos sobre directorios hijos, también podremos hacerlo sobre directorios padres. Para ello, cuando queremos ascender un nivel dentro de la estructura de directorios lo podemos hacer poniendo la siguiente orden:

C:\princip\program>CD ..

C:\princip>

Ya vimos que el directorio .. hace referencia al directorio padre.

Cada vez que ejecutemos esta orden, ascenderemos por la estructura, hasta llegar, si queremos, al directorio raíz. En este directorio esta orden no tiene sentido, ya que es el directorio padre de todos los demás.

Ejemplo.

Pongamos otro ejemplo más complicado. Supongamos que queremos llegar o situarnos en el directorio PROGRAM, pero estamos situados en el directorio SISTEMAS.

En este caso podríamos hacerlo de varias formas. Las más usuales son las siguientes:

1. Movernos por la estructura de directorios hasta llegar al deseado. Para ello, nos situamos en el directorio padre de PROGRAM, que es PRINCIP. Para ello, subiremos dos veces al directorio padre correspondiente para, posteriormente, situarnos en el directorio deseado:

C:\princip\apuntes\sistemas>CD ..

C:\princip\apuntes>CD ..

C:\princip>CD program

C:\princip\program>

2. Situarnos directamente en el directorio indicando la trayectoria absoluta:

C:\princip\apuntes\sistemas>

CD c:\princip\program

C:\princip\program>

3. Situarnos en el directorio utilizando la trayectoria relativa:

C:\princip\apuntes\sistemas> CD ..\..\program

Este caso es similar al primero, pero con una sola orden. Subimos dos niveles, para posteriormente bajar uno.

También podemos cambiar de directorio activo en otra unidad lógica, sin tener necesidad de cambiar de directorio ni unidad activa.

Supongamos que queremos ir al directorio raíz de la unidad A:, pero quedándonos en el directorio PROGRAM del disco duro:

C:\princip\program>CD a:

C:\princip\program>

De esta forma seguimos donde estábamos, pero en la unidad A:. Nos hemos situado en su directorio raíz.

Esta misma operación la podríamos haber hecho de otra forma más larga:

C:\princip\program>A:

A:\basic>CD ..

A:\>C:

C:\principal\program>

La orden **CD** seguida del símbolo \, nos sitúa directamente en el directorio raíz de la unidad en la que estemos.

Si estamos situados en PROGRAM y queremos ir al directorio raíz de la unidad C:, podremos poner lo siguiente:

C:\princip\program>CD

C:\>

O bien:

C:\princip\program>CD ..

C:\princip>CD ..

C:\>

O bien:

C:\princip\program>CD ..\..

C:\>

El resultado es el mismo. Se trata de hacer las operaciones con el mejor número de órdenes posibles y de la forma más cómoda.

MD (interno)

Crea un directorio.

SINTAXIS

MKDIR [unidad:]ruta

MD [unidad:]ruta

Ya sabemos que en toda unidad de disco existe siempre un directorio raíz. Este directorio raíz se referenciará siempre con el símbolo **\.** Este símbolo estará precedido de la letra correspondiente a la unidad lógica de la unidad en la que se esté.

El resto de directorios que tenga nuestro disco no es obligatorio. Depende de las necesidades de cada usuario, quien los crea o elimina a su gusto.

Solamente debemos tener en cuenta que la trayectoria para acceder a una estructura de directorios y subdirectorios tiene una limitación de 63 caracteres. Por eso no es conveniente crear estructuras demasiado complejas de directorios.

Para crear directorios es importante tener en cuenta dónde queremos crearlo y dónde estamos situados.

En primer lugar consideramos en qué sitio de nuestra estructura vamos a crear un nuevo directorio. Para ello podemos utilizar dos técnicas:

- Colocarnos físicamente en el directorio padre del nuevo directorio que se va a crear.
- Crear el directorio utilizando trayectorias absolutas o relativas.

Ejemplo.

Vamos a partir de la misma estructura de directorios y ficheros original (Figura 1):

Vamos a crear un nuevo directorio TEXTOS, dentro de WORD, y vamos a estar situados en el directorio PROGRAM.

Podemos crearlo situándose previamente en el directorio WORD, utilizando las trayectorias relativas o absolutas:

C:\princip\program>CD ..\docs\word

C:\princip\docs\word>MD textos

De esta forma ya hemos creado el directorio en el lugar deseado. También lo podríamos haber hecho de la otra forma:

C:\princip\program>

MD c:\princip\docs\word\textos

Utilizando la trayectoria absoluta, o bien:

C:\princip\program>MD ..\docs\word\textos

Utilizando la trayectoria relativa.

Cuando creamos nuevos directorios hemos de tener en cuenta unas consideraciones. La primera es que no pueden existir directorios ni ficheros con el mismo nombre dentro del mismo nivel de la estructura jerárquica. Es decir, dentro del directorio raíz, por ejemplo, no puede existir otro directorio llamado PRINCIP, pero sí puede existir en cualquier otro sitio. PRINCIP, como nombre de directorio, podría estar colgando de cualquier otro directorio, dentro de la misma unidad lógica o en otra unidad lógica sin ningún problema, pero nunca al mismo nivel del que ya existe.

Esto no implica que PRINCIP esté en dos sitios. Simplemente indica que hay dos directorios llamados así, pero cada uno es independiente del otro.

RD (interno)

Quita o borra un directorio.

SINTAXIS

RMDIR [/S] [/Q] [unidad:]ruta

RD [/S] [/Q] [unidad:]ruta

/S Quita todos los directorios y archivos del directorio además del mismo directorio. Se utiliza principalmente cuando se desea quitar un árbol.

/Q Modo silencioso. No pide confirmación para quitar un árbol de directorio con /S

Con esta orden podemos eliminar directorios en cualquier parte de la estructura jerárquica.

Para borrar directorios hemos de tener en cuenta que **solamente se puede borrar un directorio si está vacío.** Por lo tanto, antes de borrar realmente un directorio, tendremos que comprobar que no tiene contenido alguno; es decir, que no contiene otros ficheros o directorios.

Otra consideración importante es que nunca se puede eliminar el directorio activo.

Para borrar un directorio hay que tener como directorio activo otro diferente; puede ser el directorio padre del que se va a borrar o cualquier otro. Tampoco se puede eliminar el directorio raíz.

Ejemplo.

Supongamos que queremos borrar el directorio que anteriormente hemos creado, y seguimos situados en el mismo sitio, es decir, en PROGRAM.

Podemos hacer la operación colocándonos en el directorio padre del directorio que vamos a eliminar, o bien utilizando las trayectorias relativas o absolutas.

C:\princip\program>CD ..\docs\word

C:\princip\docs\word>RD textos

O bien:

C:\princip\program>

RD ..\docs\word\textos

O bien:

C:\princip\program>

RD c:\princip\docs\word\textos

Ejemplo.

Queremos eliminar el directorio APUNTES, y estamos situados en el directorio PROGRAM. La operación a realizar utilizando cualquiera de las formas anteriores, sería la siguiente:

C:\princip\program>RD ..\apuntes

En teoría, el directorio se tendría que haber eliminado, pero en este caso se mostrará el siguiente mensaje por pantalla:

Trayectoria no válida, no existe el directorio o el directorio no está vacío.

Evidentemente, este directorio contiene un subdirectorio que, a su vez, contiene dos archivos.

Para eliminar realmente el directorio APUNTES, tendríamos que eliminar primero el directorio SISTEMAS. Pero para eliminar el directorio SISTEMAS primero tendremos que eliminar los dos archivos que contiene (esto se hace con la orden **DELETE** que veremos más adelante).

En primer lugar, eliminaríamos los ficheros; después, pasaremos a eliminar los directorios de la siguiente forma:

C:\princip\program>RD ..\apuntes\sistemas

C:\princip\program>RD ..\apuntes

O bien:

C:\princip\program>CD ..

C:\princip>CD apuntes

C:\princip\apuntes>RD sistemas

C:\princip\apuntes>CD ..

C:\princip>RD apuntes

También podríamos hacerlo utilizando cualquier otra técnica de trayectorias relativas o absolutas.

Para eliminar un directorio hay que asegurarse de que no contiene otros directorios ni ficheros. Ya veremos que en algunas ocasiones vamos a borrar un directorio, que aparentemente está vacío, y no podemos hacerlo.

Esta situación se produce cuando el directorio tiene archivos o directorios ocultos, que son los que aparentemente no están, pero realmente sí existen.

DELTREE (externo)

Quita o borra un árbol de directorios.

ESTE COMANDO SOLAMENTE ESTA PRESENTE HASTA WINDOWS 2000. DESDE WINDOWS XP HA SIDO ELIMINADO

SINTAXIS

DELTREE [/PARÁMETROS] [UNIDAD:] [CAMINO] NOMBRE _ DIRECTORIO

/Y Elimina del comando la pregunta de petición de confirmación del borrado solicitado.

Para eliminar directorios y archivos con la orden **DELTREE,** nuestro directorio activo no puede ser uno de los que vamos a eliminar, ni podemos estar situados en ninguno contenido en él.

Con la orden **RD** solamente podemos eliminar un directorio cada vez que la ejecutamos. Es evidente que si queremos eliminar un directorio que contiene otros muchos subdirectorios que contienen otros más y que en ellos hay ficheros, la operación podría ser demasiado complicada y lenta.

Con la orden **DELTREE** se puede solucionar este problema. Este comando permite borrar directorios, incluyendo todos los archivos y subdirectorios que contenga. Es preciso tener un especial cuidado con este comando, ya que no respetará los atributos de un archivo o directorio, borrará todo lo indicado sin más. Es decir, si hay ficheros o directorios ocultos, también se borrarán, a diferencia de lo que ocurre con la orden **RD.**

Ejemplo.

Supongamos que queremos borrar toda la estructura de directorios que dependen de PRINCIP. Como hemos comentado anteriormente, para eliminar este directorio y todo lo que de él depende, solamente podremos estar situados en el directorio raíz.

C:

Situados en el directorio raíz, ejecutamos la orden:

C:\>DELTREE princip

De esta forma quedará eliminada toda la estructura, incluido el propio directorio PRINCIP.

Si ejecutamos la orden sin ningún parámetro, cada vez que se vaya a eliminar un directorio o un fichero, se nos pedirá confirmación. En este caso aparecería:

¿Eliminar directorio "……" y todos sus subdirectorios [sn]?s

¿Eliminar archivo "……" [sn]?s

Si queremos confirmar la operación pulsaremos **s** y luego **Enter;** en caso contrario pulsaremos **n** y **Enter.** Si por cualquier circunstancia queremos abortar la acción pulsaremos la combinación de teclas **Control+C.**

Si por cualquier circunstancia, introducimos una trayectoria incorrecta o un nombre de directorio que no existe, la orden no proporcionará ningún mensaje de error. Simplemente saldrá otra vez el símbolo del sistema en el directorio activo.

Esta orden también puede utilizarse para eliminar varias estructuras de directorios de una vez.

Ejemplo.

Supongamos que queremos eliminar los directorios DOCS y APUNTES, y mantener el resto.

Previamente nos tendríamos que situar en una trayectoria en la que no esté contenido ningún fichero ni directorio de la estructura que vamos a eliminar, por ejemplo, podríamos situarnos en el directorio PRINCIP.

La orden sería la siguiente:

C:\PRINCIP>

DELTREE c:\princip\docs c:\princip\apuntes

Se eliminarían estas dos estructuras de directorios y todos sus archivos. Por supuesto, se pide confirmación antes de proceder al borrado real.

Si utilizamos la orden **DELTREE** con el parámetro **/Y,** estaremos anulando la confirmación de borrado de cada fichero y directorio.

MOVE (externo)

Mueve archivos y cambia el nombre a archivos y directorios.

SINTAXIS

Mover uno o más archivos:

MOVE [/Y | /-Y] [unidad:][ruta]nombrearchivo1[,...] destino

Cambiar el nombre a un directorio:

MOVE [/Y | /-Y] [unidad:][ruta]nombredirectorio1 nombredirectorio2

[unidad:][ruta]nombrearchivo1 Especifica la ubicación y el nombre del archivo o archivos que quiere mover.

destino Específica la nueva ubicación del archivo. El destino puede constar de una letra de unidad y dos puntos, un nombre de directorio o una combinación. Si mueve solo un archivo, también puede incluir un nombre de archivo si quiere cambiar el nombre del archivo cuando lo mueva.

[unidad:][ruta]nombredirectorio1 Especifica el directorio al que desea cambiar el nombre.

nombredirectorio2 Específica el nuevo nombre del directorio.

/Y Suprime la petición de confirmar si quiere sobrescribir un archivo de destino ya existente.

/-Y Aparecerá la petición para confirmar si desea sobrescribir un archivo de destino ya existente.

El modificador **/Y** puede estar presente en la variable de entorno **COPYCMD.**

Esto puede sobrescribirse con **/-Y** en la línea de comandos. Lo predeterminado es pedir la sobrescritura a no ser que el comando **MOVE** sea ejecutado dentro de una secuencia de comandos de un archivo de proceso por lotes.

Con esta orden podemos cambiar el nombre de un directorio por otro. Es evidente que el directorio tiene que existir y da igual dónde esté situado.

Con este comando no se puede renombrar el directorio activo, pero sí cualquier otro. Lo normal es ejecutar esta orden (especialmente diseñada para cambiar la ubicación o el nombre a ficheros y no cambiar el nombre a directorios) desde el directorio padre del directorio que se quiere renombrar.

Ejemplo.
Vamos a cambiar el nombre al directorio PROGRAM. El nuevo nombre será PROGRAM1. Para ello, lo primero que haremos será situarnos en el directorio padre: **C:\princip>MOVE program program1**

MOVE (externo) Muestra de forma gráfica la estructura de carpetas de una unidad o ruta.
SINTAXIS TREE [unidad:][ruta] [/F] [/A]
/F Muestra los archivos de cada carpeta. **/A** Usa ASCII en lugar de caracteres extendidos.

El comando **TREE** muestra la estructura de directorios y subdirectorios contenida en cada uno de ellos. Si se especifica un directorio, el comando emitirá la estructura que cuelgue de éste, no los que puedan existir por encima de él.

8.8. Documentación y ayuda en entorno comando.

http://commandwindows.com/command1.htm

La documentación que normalmente aporta la mayoría de los sistemas operativos puede ser de dos tipos:

- **Documentación externa.** Son los manuales que incorpora el sistema operativo y que sirven al usuario para realizar la instalación del sistema operativo, consultar dudas respecto a prestaciones y características del mismo, etcétera.
- **Documentación interna.** El sistema operativo también aporta un sistema interactivo de información referente a los comandos y órdenes deseadas. Este sistema muestra la información, parámetros y formas de utilizar las diferentes órdenes.

El sistema de ayuda se obtiene tecleando el comando y a continuación **"/?",** También puede obtenerse información a través del mandato **HELP.**

- **Ficheros TXT.** Es habitual que Windows y otros sistemas operativos incluyan dentro de sus archivos algunos con la extensión .txt . Estos ficheros pueden ser visualizados o impresos, y nos mostrarán información más completa sobre el uso o parámetros de comandos o programas.

8.9. Teclas y caracteres especiales en entorno comando de Windows.

Especial interés tienen algunas teclas incluidas en el teclado de escritura o teclado QWERTY:

- **Tecla Intro,** también llamada **Return** *o* **Enter.** Con esta tecla lo que hacemos es procesar, una vez que la pulsamos, la orden introducida tras el símbolo del sistema. Mientras que la tecla no se pulse, el sistema operativo no realizará ninguna acción, quedándose en espera.

Esta tecla también se localiza en el teclado numérico en la parte inferior derecha.

- **Teclas Control y Alt.** Estas teclas, como tales, no tienen una función determinada. Para que tengan efectividad, tendrán que ser pulsadas simultáneamente con alguna otra tecla.

Ejemplo.
Por ejemplo, la combinación de teclas **Control+C** provoca que se interrumpa determinada acción que se esté ejecutando. Por ejemplo, la ejecución de un fichero por lotes.

- La tecla **Alt** tiene varias funciones, pero una de las fundamentales es la de mostrar en pantalla el carácter correspondiente a un código ASCII concreto. Para ello, pulsaremos el tecla y, sin soltarla, pulsaremos la combinación decimal del código ASCII del carácter que queremos visualizar. En cuanto soltemos la tecla **Alt** aparecerá en pantalla el carácter deseado.

Una combinación importante es la formada por las teclas **Control+Alt+Supr**. Con ello, realizamos un **Reset**: apagamos el sistema; es como si desenchufásemos el ordenador y volviésemos a enchufarlo a la red eléctrica. Esta combinación se utiliza cuando el ordenador se ha quedado bloqueado al ejecutar alguna acción concreta o durante la ejecución de un proceso o programa.

- **Teclas de función.** Estas teclas son las que aparecen en la parte superior del teclado. Se identifican fácilmente, ya que están formadas por una **F** y por un número que suele ir del 1 al 12.

No suelen tener utilidad en la ejecución de órdenes del sistema operativo, pero sí funciones asociadas de programas concretos. Normalmente, se utilizan en procesadores de texto, hojas de cálculo, etc., asociando a cada tecla una acción. Por ejemplo, **F1** es imprimir; **F2** es guardar un archivo, etcétera.

- **Tecla Alt Gr.** Esta tecla tiene que usarse en combinaciones con otras teclas para que tenga utilidad. Suele utilizarse para poder visualizar los caracteres indicados en la parte frontal de algunas letras o parte inferior derecha de la misma.

Si queremos obtener el carácter inferior de la tecla bastará con pulsarla. Si queremos obtener el carácter superior, tendremos que pulsar la tecla **Shift** (mayúsculas). Si queremos obtener el carácter frontal, tendremos que pulsar **Alt Gr**, la combinación **Control+Alt+Tecla** (en algunos teclados) o la combinación **Shift+Alt+Tecla** (en otros teclados).

8.10. Nombre de los dispositivos en Windows. Discos, impresoras, consola, puertos serie, etcétera.

La información que maneja el ordenador son datos, los cuales le son suministrados mediante los dispositivos de entrada y, gracias a la ejecución de un programa u orden concreta; son procesados para obtener los datos de salida.

Los programas son los que se encargan de ejecutar la información que se le ha suministrado, pero en ningún momento nos tiene que preocupar de qué forma realiza este proceso.

Los dispositivos periféricos pueden ser **unidireccionales.** Son los que envían o reciben información. Pueden ser de entrada o de salida, pero no de entrada/salida (teclado, monitor).

También puede haber dispositivos **bidireccionales,** que son aquellos que pueden recibir y enviar información. Son los dispositivos de entrada/salida (discos duros, módems).

Ya sabemos cuáles son los diferentes periféricos que puede manejar un ordenador; concretamente, en Windows, son cinco:

- **Teclado y monitor. Dispositivos CON.** Son unidireccionales y los denominados dispositivos estándares de entrada y de salida. En Windows a estos dispositivos se les asigna un nombre **CON**, que es la abreviatura de **CON**sole.
- **Puertos paralelo. Dispositivos LPTn.** Los puertos o interfaces paralelo son dispositivos bidireccionales que permiten enviar y recibir datos. Su nombre proviene de las palabras ***L****ine* ***P****rin****T****er.* Son los dispositivos paralelo o **centronics** vistos en la unidad anterior. Se denomina puertos paralelos porque son capaces de transmitir información de byte en byte, es decir, los 8 bits que forman el byte se envían a la vez. Estos puertos suelen usarse para conectar impresoras al ordenador. El sistema operativo es capaz de reconocer tres puertos paralelos distintos: **LPT1, LPT2** y **LPT3,** que indican, respectivamente, primero, segundo y tercer puerto paralelo. Al puerto LPT! se suele conectar la impresora. Este puerto también recibe el nombre de **PRN** que viene de ***PR****i****N****ter.*
- **Puertos serie. Dispositivos COMn.** Son dispositivos bidireccionales; pero, al contrario de los anteriores, no permiten transferir información byte a byte. Estos puertos transmiten los bits que forman un byte secuencialmente, uno detrás de otro

  A estos dispositivos suele conectarse la mayoría de los periféricos de entrada, salida o de entrada/salida.

  En puerto estándar serie que utilizan los ordenadores es el puerto denominado **RS-232-C,** que puede presentarse físicamente en dos formatos: los de 25 pines (conectores), usado en los XT, y los de 9, usado en los AT. El funcionamiento de ambos es el mismo.

  El MS-DOS, a partir de la versión 3.3. es capaz de soportar hasta cuatro puertos serie: **COM1**, **COM2**, **COM3** y **COM4.** Al puerto COM1 también se le denomina **AUX.**
- **Dispositivo nulo. NULL.** Este dispositivo puede considerarse como un dispositivo físico o nulo. De esta forma, cuando queramos realizar una transferencia de información a **ningún sitio,** utilizaremos el dispositivo NULL. Si el dispositivo funciona como entrada, no habrá entrada, y si funciona como salida, no habrá salida.

  En realidad, se trata de simular la transferencia o recepción de información.
- **Reloj CLOSCK$**. Es un dispositivo que el sistema reconoce al arrancar el sistema. Este reloj puede ser un reloj en tiempo real cargado en la **CMOS** del ordenador, o un reloj de intervalos que se cargará en memoria al encender el ordenador.

  Cuando veamos, en la siguiente unidad, las órdenes de manejo de ficheros y directorios, y el redireccionamiento, entenderemos la utilidad de estos nombres de dispositivo que el sistema asigna.

8.11. Órdenes de manejo de ficheros.

http://commandwindows.com/command1.htm

A. Introducción.

Ya sabemos que hay dos tipos de ficheros fundamentales en casi todos los sistemas operativos:

- **Ficheros o archivos regulares.**
- **Directorios.**

Los ficheros o archivos regulares suelen ser de dos tipos:

- **Ficheros ejecutables.** Son aquellos que se pueden ejecutar directamente escribiendo su nombre, sin extensión. Solamente son ejecutables los ficheros que tienen extensión **COM, EXE** o **BAT.**
- **Ficheros de datos.** No son ejecutables y contienen información de cualquier otro tipo. En la Unidad 7 hemos visto las extensiones de algunos de estos ficheros

B. Caracteres comodines.

Se utilizan para identificar varios ficheros de una sola vez. Se llaman así porque pueden sustituir a un carácter o a varios en los nombres de fichero.

Los caracteres comodines en Windows son los siguientes:

- **Carácter ?.** Representa cualquier carácter válido en el nombre de un fichero. Solo sustituye o representa unos de ellos.
- **Carácter *.** Representa uno o más caracteres válidos en el nombre de un fichero. Sustituye caracteres por el principio, por el final o por el centro del nombre de archivo especificado.

Ejemplo.

Supongamos que tenemos la siguiente estructura de ficheros dentro del directorio raíz:

```
\
 |
 |____AUTOEXE.BAT
 |_____ COMMAND.COM
 |_____ PROGRAM.SYS
 |_____ PROGRAM.BAK
 |
 PROGRAM.BAS
```

Y supongamos que ponemos la orden

C:\>DIR program.ba?

En este caso se visualizarían todos los archivos llamados PROGRAM cuya extensión empezase por BA y cuyo tercer carácter fuese cualquiera. Se visualizarían los ficheros PROGRAM.BAS y PROGRAM.BAK.

Si ponemos:

C:\>DIR program.B?S

solamente se visualiza el fichero PROGRAM.BAS, ya que el comodín solamente sustituye el

carácter central de la extensión manteniendo el resto.

Si ponemos:

C:\>DIR program.*

estaremos visualizando todos los archivos cuyo nombre es PROGRAM y con cualquier extensión; es decir, PROGRAM.BAS, PROGRAM.BAK y PROGRAM.SYS.

Al ejecutar:

C:\>DIR program.B*

se visualizarán todos los archivos cuya extensión tenga una B como primer carácter y cualesquiera otros dos. Solo se visualizarán PROGRAM.BAK y PROGRAM.BAS

Otro ejemplo es:

C:\>DIR pro*.*s

En este caso se visualizan los ficheros PROGRAM.BAS y PROGRAM.SYS.

El funcionamiento es el mismo con los nombres de fichero que con las extensiones. Un **?** sustituye a un solo carácter y un * a todos los anteriores, primeros o últimos.

A continuación se muestran unas plantillas típicas para el manejo de nombres de ficheros:

- ***.EXE.** Plantilla para todos los ficheros con extensión EXE.
- **?????.BAT.** Plantilla para todos los ficheros con un nombre de cinco caracteres y extensión BAT.
- ***.??.** Plantilla para todos los ficheros con cualquier nombre y cualquier longitud, y con solo dos caracteres de extensión.
- ***.*** Plantilla para todos los ficheros.
- **????????.???.** Plantilla para todos los ficheros.

En Windows, la mayoría de las órdenes de manejo de ficheros que pueden incluir trayectorias aceptan caracteres comodín, a excepción de la orden TYPE.

C. Órdenes de manejo de ficheros.

TYPE (interno)
Muestra el contenido de uno o más archivos de texto.
SINTAXIS
TYPE [unidad:][ruta]archivo

Muestra el contenido de un archivo por la salida estándar (pantalla). Generalmente solo aparecerán legibles los archivos de texto. Este comando no permite el uso de comodines.

Ejemplo.

Si queremos visualizar el contenido del fichero TEMA1.DOC del directorio SISTEMAS, y estamos situados en SISTEMAS, la orden será la siguiente:

C:\princip\apuntes\sistemas>TYPE tema1.doc

Si estamos situados, por ejemplo, en PROGRAM, para visualizar el mismo fichero, podemos actuar de dos modos: situarnos en el directorio en el que está el archivo o fichero que se va a visualizar (como antes) o utilizar las trayectorias relativas o absolutas:

C:\princip\program>TYPE ..\apuntes\sistemas\tema1.doc

O bien:

C:\princip\program>TYPE c:\princip\apuntes\sistemas\tema1.doc

PRINT (externo)

Imprime un archivo de texto.

SINTAXIS

PRINT [/D: dispositivo] [[unidad:][ruta]archivo[...]]

/D: dispositivo Especifica un dispositivo de impresión.

Con esta orden podemos imprimir el contenido de un fichero.

El funcionamiento es similar a la orden **TYPE,** con la diferencia de que se pueden imprimir varios ficheros a la vez. Para ello utilizaremos los caracteres comodín necesarios.

Un ventaja de este comando es que pone los trabajos a imprimir en una cola de espera retornando rápidamente el control del sistema al usuario; es decir, el usuario puede seguir trabajando mientras se imprimen los trabajos.

Ejemplo.

Supongamos que queremos imprimir todos los archivos del directorio SISTEMAS y estamos situados en el directorio raíz:

C:\>PRINT c:\princip\apuntes\systemas*.*

Tras esta orden, el sistema operativo indicará qué trabajo se está imprimiendo y cuál de ellos está en cola de impresión. Funciona con un pequeño *spool*.

Si queremos cancelar los trabajos de impresión antes de que se acaben de imprimir, ejecutaremos la orden:

C:\>PRINT /T

RENAME (interno)

Cambia el nombre de uno o más archivos.

SINTAXIS

RENAME [unidad:][ruta]archivo1 archivo2.

REN [unidad:][ruta]archivo1 archivo2.

No puede especificar una nueva unidad o ruta para el archivo de destino.

Cambia el nombre a uno o más ficheros. Este comando admite el uso de comodines. No puede existir previamente en el directorio otro fichero con el mismo nombre que se dará al fichero que se va a renombrar.

Si existe trayectoria para el fichero original, ésta será la del nuevo fichero. La trayectoria de destino del nuevo fichero no existe. Es decir, el nuevo fichero se crea en la trayectoria especificada del primero.

Ejemplo.

Vamos a cambiar el nombre del fichero Tema1.doc por Nuevo.doc, del directorio SISTEMAS. Estamos en el directorio raíz:

C:\>REN c:\princip\apuntes\sistemas\tema1.doc nuevo.doc

También podemos renombrar un bloque o conjunto de ficheros. Supongamos que a todos los ficheros del directorio SISTEMAS les queremos cambiar la extensión DOC por otra nueva llamada BAK:

C:\>REN c:\princip\apuntes\sistemas*.doc *.bak

El directorio destino de los nuevos ficheros tiene que ser el mismo en el que están los originales. La siguiente orden, por ejemplo, no es correcta:

C:\>REN c:\princip\fichero.txt c:\princip\program\nuevo.txt

Si queremos hacer algo parecido solamente tenemos una solución: copiar el fichero en el nuevo directorio con el nuevo nombre y eliminar el fichero original.

COPY (interno)

Copia uno o más archivos en otra ubicación.

SINTAXIS

COPY [/V] [/N] [/Y | /-Y] [/Z] [/A | /B] origen [/A | /B] [+ origen [/A | /B] [+ ...]] [destino [/A | /B]]

origen Especifica el archivo o archivos que deben copiarse.

/A Indica un archivo de texto ASCII.

/B Indica un archivo binario.

/D Permite al archivo de destino que se cree descifrado.

Destino Especifica el directorio y el nombre de archivo de los nuevos archivos.

/V Verifica que los nuevos archivos se escriben correctamente.

/N Si es posible, usa un nombre de archivo corto al copiar un archivo cuyo nombre no tiene el formato 8.3.

/Y Suprime la petición de confirmación cuando se va a sobrescribir un archivo destino existente.

/-Y Realiza la petición de confirmación cuando se va a sobrescribir un archivo destino existente.

/Z Copia archivos de red en modo reiniciable.

El modificador **/Y** puede estar prestablecido en la variable de entorno **COPYCMD.** Esto puede anularse con el modificador **/-Y** en la línea de comando. Está predeterminado el pedir la confirmación del usuario antes de sobrescribir, excepto si comando **COPY** se ejecuta desde un archivo de comandos por lotes.

Para anexar archivos, especifique un único archivo de destino, pero varios archivos de origen (usando caracteres o el formato archivo1+arc hivo2+archivo3).

Copia uno o más archivos a otro lugar. Si no se especifica trayectoria de destino, la copia se realizará en el directorio y unidad predeterminados (activos) y tendrá el mismo nombre que el archivo original. Si no se indica el camino, se copiarán los ficheros que se encuentren en la unidad y directorio actuales. Este comando permite la utilización de comodines.

Para el manejo correcto de esta orden, es necesario tener en cuenta varias cosas:

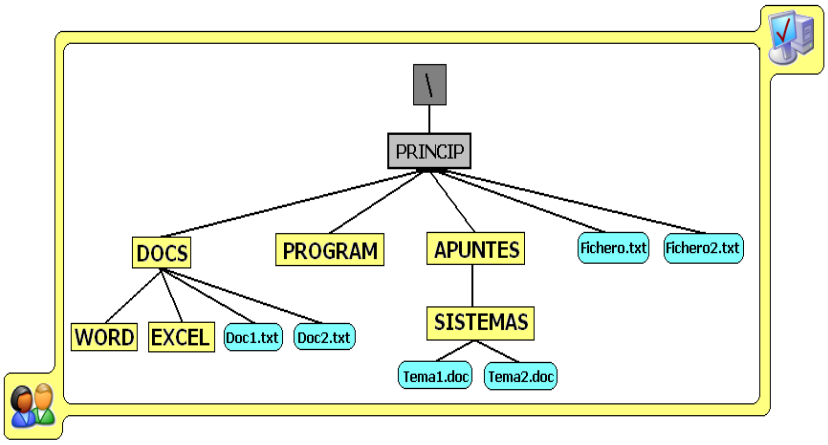
- Dónde estamos situados.
- Dónde se encuentra el fichero que se va a copiar y su nombre.
- Dónde queremos copiar el fichero.
- Cuál es el nombre con el que queremos copiar el fichero, con el mismo o con otro.

Cuando ejecutamos esta orden, tenemos que tener en cuenta que:

- Si en el directorio de destino existe un fichero con el mismo nombre, se reemplaza por el nuevo fichero. En las últimas versiones de Windows, se nos pedirá confirmación de la copia.
- Esta orden no sirve para copiar ficheros ocultos ni ficheros de sistema.
- Con la orden **COPY** se pueden copiar varios ficheros en uno solo; es decir, se pueden concatenar varios archivos o ficheros en otro diferente.

Ejemplo.

Recordemos la estructura de archivos y directorios de nuestros ejemplos:



Supongamos que queremos copiar el fichero tema1.doc del directorio SISTEMAS al directorio PROGRAM.
Estamos situados en WORD:

C:\princip\docs\word>COPY c:\princip\apuntes\sistemas\tema1.doc C:\princip\program

Como se puede apreciar, el nombre del fichero de destino no se ha puesto. Esto es debido a que queremos copiar el fichero con el mismo nombre. En este caso basta con especificar solamente la trayectoria de destino.

Si lo quisiéramos copiar con otro nombre, por ejemplo tema1.bak, la orden sería la siguiente:

C:\princip\docs\word>COPY c:\princip\apuntes\sistemas\tema1.doc C:\princip\program\tema1.bak

Cuando no especificamos destino, el fichero se copia en el directorio y unidad activos:

C:\princip\docs\words>COPY c:\princip\apuntes\sistemas\tema1.doc tema1.bak

El fichero tema1.doc se copia en el directorio WORD con el nombre tema1.bak.

Si tenemos como unidad y directorio activo la trayectoria en la que se encuentra el fichero de origen, no hay que especificar más que la trayectoria y nombre de destino del fichero:

C:\princip\apuntes\sistemas>COPY tema1.doc C:\princip\program\tema1.bak

En este caso copiamos el fichero team1.doc en el directorio PROGRAM, con el nombre tema1.bak. Si ejecutamos la siguiente orden:

C:\princip\apuntes\sistemas>COPY tema1.doc tema1.bak
Copiamos el fichero en el mismo directorio con otro nombre.

Todo lo explicado en cuanto al manejo de trayectorias es válido para esta orden.

También se puede copiar un conjunto de archivos en otro directorio. Para ello utilizaremos los caracteres comodín.

Ejemplo.

1. Supongamos que queremos copiar con el mismo nombre todos los ficheros del directorio SISTEMAS en el directorio PROGRAM. Estamos situados en el directorio donde están los archivos que vamos a copiar:
   C:\princip\apuntes\sistemas>COPY *.* C:\princip\program
   También podemos copiar todos los archivos, de una o varias extensiones, en otros con otras extensiones.
2. Supongamos que queremos copiar los ficheros del directorio SISTEMAS con extensión DOC en el directorio PROGRAM, pero con la extensión BAK y con el mismo nombre:

   C:\princip\apuntes\sistemas>COPY *.doc c:\princip\program*.bak

Para utilizar adecuadamente la orden **COPY** tenemos que considerar el lugar en el que se encuentra el fichero o ficheros que se van a copiar, el nombre de los mismos, el lugar en el que los queremos copiar y dónde estamos situados.

Si estamos situados en el directorio donde están los archivos de origen, no hay que especificar la trayectoria de origen. Si estamos en la trayectoria de destino, no hay que especificar la trayectoria de destino. Si no estamos ni en la trayectoria de origen ni en la de destino, hay que especificar ambas.

En cuanto a los nombres de los archivos que vamos a copiar, siempre hay que indicar el nombre del archivo o archivos de origen (si es necesario, utilizando comodines). El nombre del archivo o fichero de destino es necesario especificarlo siempre que queramos que sea diferente.

Para utilizar las trayectorias, podemos utilizar las técnicas de trayectorias absolutas o relativas indistintamente.

Ejemplo.

Supongamos que queremos copiar todos los archivos del directorio SISTEMAS en un solo archivo llamado fusión.doc en el directorio raíz. Estamos situados en SISTEMAS:

C:\princip\apuntes\sistemas>COPY tema1.doc + tema2.doc c:\fusion.doc

El primer caso se utiliza para concatenar archivos que tengan diferente nombre y extensiones diferentes. Pero, en nuestro ejemplo, es más cómoda la siguiente opción, ya que todos los archivos tienen la misma extensión. El resultado es el mismo.

C:\princip\apuntes\sistemas>COPY *.doc c:\fusion.doc

Por último, la orden **COPY** también puede utilizar los nombres de dispositivo como origen o destino de los ficheros.

Ejemplo.

Supongamos que queremos imprimir el fichero tema1.doc. Podemos ejecutar cualquiera de estas órdenes:

C:\princip\apuntes\sistemas>PRINT tema1.doc

C:\princip\apuntes\sistemas>TYPE tema1.doc>prn

Más adelante veremos el redireccionamiento.

C:\princip\apuntes\sistema>COPY tema1.doc prn

En este último caso estamos copiando el fichero en el dispositivo PRN, que es la impresora. De esta forma el documento sale impreso. También podíamos haber puesto LPT1 en vez de PRN.

Si queremos visualizar un fichero, utilizaremos la orden **TYPE,** pero también podemos copiarlo al dispositivo de pantalla:

C:\princip\apuntes\sistemas>COPY tema1.doc con

El resultado es prácticamente el mismo.

También podemos utilizar, principalmente, el dispositivo CON como origen de un fichero de datos que queremos copiar.

Ejemplo.

Copiamos, desde pantalla, un archivo llamado fichero.txt en el directorio raíz:

C:\COPY CON fichero.txt

Este comando es una prueba de cómo se puede copiar un archivo desde la pantalla al disco Ctrl+Z+Enter

1 Archivos(s) copiado(s)

C:\>

Este comando así utilizado permite introducir como origen datos desde el teclado a través de la pantalla (en nuestro caso, el texto en cursiva). Finalizada la introducción del texto, se pulsa la combinación de teclas **Control+Z** y el contenido tecleado se copia como archivo en la trayectoria especificada y con el nombre especificado. En nuestro caso, el fichero de destino es fichero.txt en el mismo directorio activo en que estamos.

Ejemplo.

Supongamos que queremos imprimir un texto en la impresora. Lo normal sería:

1. Crear el fichero con un editor de texto (lo veremos en la unidad 6).

2. Ejecutar el comando **PRINT, TYPE** con redireccionamiento a **PRN** o **COPY** y, como destino, el dispositivo impresora.

Pero podemos hacer lo siguiente:

C:\>COPY CON PRN

Este es el texto que se imprimirá cuando pulsemos las teclas de fin de copia **Ctrl+Z+Enter.**

1 Archivo(s) copiado(s)

C:\>

El origen es la consola (CON), y el destino es la impresora (PRN). Lo único que ocurre es que no queda copia del archivo en ningún sitio, a excepción de la copia impresa.

MOVE (externo)

Mueve archivos y cambia el nombre a archivos y directorios.

SINTAXIS

Mover uno o más archivos:

MOVE [/Y | /-Y] [unidad:][ruta]nombrearchivo1[,...] destino

Cambiar el nombre a un directorio:

MOVE [/Y | /-Y] [unidad:][ruta]nombredirectorio1 nombredirectorio2

[unidad:][ruta]nombrearchivo1 Especifica la ubicación y el nombre del archivo o archivos que quiere mover.

destino Especifica la nueva ubicación del archivo. El destino puede constar de una letra de unidad y dos puntos, un nombre de directorio o una combinación. Si mueve solo un archivo, también puede incluir un nombre de archivo si quiere cambiar el nombre del archivo cuando lo mueva.

[unidad:][ruta]nombredirectorio1 Especifica el directorio al que desee cambiar el nombre.

nombredirectorio2 Especifica el nuevo nombre del directorio.

/Y Suprime la petición de confirmar si quiere sobrescribir un archivo de destino ya existente.

/-Y Aparecerá la petición para confirmar si desea sobrescribir un archivo de destino ya existente.

El modificador /Y puede estar presente en la variable de entorno **COPYCMD.**

Esto puede sobrescribirse con **/-Y** en la línea de comandos. Lo predeterminado es pedir la sobrescritura a no ser que el comando **MOVE** sea ejecutado dentro de una secuencia de comandos de un archivo de proceso por lotes.

Mueve uno o varios archivos de un directorio a otro. Este comando también permite la utilización de comodines cuando queremos mover varios archivos a la vez.

La diferencia de **MOVE** respecto a **COPY** es clara: **MOVE** solamente mantiene una copia del archivo en la estructura. **COPY** permite mantener dos copias, el original y el nuevo archivo.

El uso de esta orden es similar a la orden **RENAME,** con la excepción de que la trayectoria de destino en **REN** no se puede especificar. Con la orden **MOVE** sí se puede.

Ejemplo.

Vamos a mover el fichero tema1.doc del directorio SISTEMAS al directorio APUNTES con el nombre tema1.bas. Estamos situados en APUNTES:

C:\princip\apuntes>MOVE sistemas\tema1.doc tema1.bas

Aquí podemos apreciar, además, otra forma de referenciar la trayectoria de un directorio hijo. Con esta orden está permitido el uso de comodines.

C:\princip\apuntes>MOVE c:\princip\apuntes\sistemas\tema1.doc C:\nuevo\tema1.bak /Y

Movemos el fichero al nuevo directorio que no existe, pero previamente, sin ningún tipo de confirmación, se crea el fichero nuevo.

DEL (interno)

Elimina uno o más archivos

SINTAXIS

DEL [/P] [/F] [/S] [/Q] [/A[[:]atributos]] nombres

ERASE [/P] [/F] [/S] [/Q] [/A[[:]atributos]] nombres

nombres Especifica una lista de uno o más archivos o directorios. Se puede utilizar comodines para eliminar varios archivos. Si se especifica un directorio todos sus archivos se eliminarán.

/P Pide confirmación antes de eliminar cada archivo.

/F Fuerza la eliminación de archivos de solo lectura.

/S Elimina archivos especificados en todos los subdirectorios.

/Q Modo silencioso. No pide confirmación con comodín global.

/A Selecciona los archivos que se van a eliminar basándose en los atributos.

atributos

- R Archivos de solo lectura.
- S Archivos de sistema.
- H Archivos ocultos.
- A Archivos preparados para almacenamiento.
- - Prefijo de exclusión.

Si las extensiones de comando están activadas, **DEL** y **ERASE** cambian de la siguiente manera:

La semántica que se muestra para el modificador **/S** está invertida, de tal modo que le muestra solamente los archivos eliminados y no los que no se encontraron.

Con esta orden y con **ERASE** podemos eliminar archivos dentro de la estructura jerárquica.

El funcionamiento es sencillo, ya que con esta orden se puede eliminar un solo fichero o varios (utilizando los comodines).

No se puede utilizar para eliminar directorios; solamente para borrar los archivos regulares de un directorio.

Cuando se elimina un fichero, en principio se elimina de forma definitiva, aunque hay utilidades que sirven para recuperarlo.

Con la orden **DEL** no se pueden eliminar archivos ocultos. Para eliminar ficheros ocultos es necesario quitar los atributos necesarios (en el caso que veremos a continuación) y después eliminar el fichero de forma normal.

Ejemplo.

Supongamos que queremos eliminar del directorio SISTEMAS el fichero tema1.doc. Solamente tenemos que tener en cuenta dónde estamos situados, el nombre del fichero y dónde se encuentra el fichero que se va a borrar.

Si estamos en el directorio raíz, la operación que se debe realizar sería la siguiente:

C:\>DEL princip\apuntes\sistemas\tema1.doc

Si estamos situados físicamente en el directorio en el que se encuentra el archivo, no tendremos que hacer nada especial:

C:\princip\apuntes\sistemas>DEL tema1.doc

Si nos encontramos en cualquier otro sitio, basta con indicar la trayectoria relativa o absoluta para realizar la operación de borrado.

Con esta orden, el uso de comodines está totalmente justificado.

Si ponemos la siguiente orden:

C:>DEL fich *.b?t

Estaremos eliminando todos aquellos ficheros cuyos cuatro primeros caracteres del nombre son *fich* y el resto, hasta los ocho caracteres, cualquier otra cosa. Se eliminarán los archivos que cumplan esta plantilla para el nombre y cuya extensión empiece por *b* y acabe por *t*, independientemente del carácter central.

ATTRIB (externo)

Muestra o cambia los atributos de un archivo

SINTAXIS

ATTRIB [+R | -R] [+A | -A] [+S | -S] [+H | -H] [unidad:][ruta] [nombre-archivo] [/S [/D]]

+ Establece un atributo.

- Borra un atributo.

R Atributo de solo lectura del archivo.

A Atributo de archivo de almacenamiento.

S Atributo de archivos del sistema.

H Atributo de archivo oculto.

[unidad:][ruta][nombre-archivo]

Especifica el archivo o archivos que serán afectados por **ATTRIB.**

/S Procesa archivos que coinciden en la carpeta actual y todas las carpetas.

/D Procesa carpetas.

Ya sabemos que cada fichero tiene un nombre y puede tener o no una extensión. Todos los archivos, en todos los sistemas, tienen también atributos.

Cada fichero o directorio puede tener cuatro atributos en Windows. Son los siguientes:

- **LECTURA (R)** ***(Read Only).*** Se especifica con una *R*. Un fichero que tenga este atributo no se podrá borrar ni modificar, solamente se podrá visualizar su contenido; es decir, solamente se podrá leer.

  Si intentamos eliminar un fichero con la orden **DEL,** no será posible, produciéndose un error con el mensaje: *Acceso denegado.*

- **ARCHIVO (A)** ***(Archive).*** Se especifica con una *A*. Este atributo sirve para saber si se ha modificado o no un fichero. Cuando se crea un archivo, por defecto, se le asigna el atributo de archivo.

  Supongamos que creamos un archivo. Se le asignará el atributo de archivo. Si, a continuación, eliminamos este atributo, el fichero seguirá siendo el mismo: podremos leerlo, imprimirlo, etcétera.

  En este caso, el archivo seguirá sin tener el atributo de archivo hasta que modifiquemos su contenido. Cuando lo hagamos, volverá a aparecer el atributo indicando precisamente que ese archivo ha sido modificado.

  Con este atributo podemos saber qué ficheros han sido modificados o no. Es evidente que para saber esto, en primer lugar tendremos que quitarlo y cuando vuelva a aparecer sabremos que el archivo ha sido modificado.

- **SISTEMA (S)** ***(System).*** Se especifica con una *S*. Establece el archivo como archivo de sistema. Hay dos archivos fundamentales con el atributo de sistema: MS-DOS.SYS e IO.SYS. Estos forman el núcleo del sistema operativo MS-DOS.

  Normalmente, son los únicos que tienen este atributo, aunque puede ocurrir que el usuario se lo asigne a cualquier fichero por su cuenta.

  Cuando un archivo es de sistema, es como si estuviera oculto; es decir, no se ve con la orden **DIR** sin parámetros, no se puede eliminar, no se puede copiar, etcétera.

- **OCULTO (H)** ***(Hidden)***. Se Especifica con una *H*. Provoca que el archivo permanezca oculto durante las operaciones normales con archivos.

  Con la orden **DIR** no se visualizarán los archivos que tengan este atributo, a menos que la utilicemos con el parámetro **/AH.** Los archivos que contengan este atributo no se pueden borrar con **DEL** ni se pueden copiar con **COPY.**

A continuación, vamos a ver las operaciones que se pueden realizar con los atributos sobre uno o varios ficheros:

1. Visualizar los atributos de un fichero.

La orden **ATTRIB** se puede utilizar de diferentes formas:

- **ATTRIB.** Puesta sin parámetros, muestra por pantalla los atributos de todos los ficheros del directorio activo.

- **ATTRIB nombre_de_fichero.** Se visualizan los atributos del fichero especificado. Es evidente que el fichero en cuestión no tiene que estar necesariamente en el directorio activo. Por ello, podemos utilizar la trayectoria absoluta o relativa para diferenciarlo.

- **ATTRIB varios_ficheros.** Esta orden, para visualizar los atributos de un conjunto o de todos los ficheros, necesita obligatoriamente el uso de los comodines. No es posible poner **ATTRIB C:\,** ya que no saldrá nada. Hay que poner **ATTRIB C:*.*,** o

cualquier otra plantilla para poder ver los atributos de todos o de un conjunto de archivos.

Si se utiliza el parámetro **/S,** se visualizarán además los atributos de los ficheros que se ajusten a la plantilla y que estén en subdirectorios que cuelguen del directorio especificado en la orden.

Ejemplo.

Vamos a visualizar los atributos de los archivos del directorio SISTEMAS y de los directorios que de él dependen. Estamos situados en APUNTES:

C:\princip\apuntes>ATTRIB sistemas*.* /S

Otra forma de visualizar los atributos de lectura de todos los archivos de un directorio concreto consiste en utilizar la orden **DIR** con el parámetro que incorpora para dicha función:

C:\princip\apuntes>DIR sistemas*.* /AR

La diferencia de esta orden con la de la anterior es que con la orden **DIR** solamente visualizamos los ficheros del directorio especificado. Con la anterior orden y con el parámetro **/S,** visualizamos además los atributos de los ficheros del resto de directorios.

También con la orden **DIR** podemos visualizar los archivos que tienen determinados atributos.

Ejemplo.

C:>DIR c:\princip /AS-H

Con esta orden visualizaremos qué archivos del directorio PRINCIP tienen atributos de sistema y no lo tienen de oculto.

1. **Dar o quitar atributos a ficheros.**

Los atributos se ponen de la misma forma que se quitan. Solamente el símbolo + o – indicará, respectivamente, que el atributo se asigna o se elimina a un archivo o a un conjunto de archivos.

Ejemplo.

Supongamos que queremos asignar los atributos de oculto y lectura al fichero tema1.doc, del directorio SISTEMAS:

C:\princip\apuntes\sistemas>ATTRIB +h +s tema1.doc

Si lo que queremos es quitar, por ejemplo, el atributo de archivo al mismo fichero:

C:\princip\apuntes\sistemas>ATTRIB –a tema1.doc

Esta misma operación se puede hacer para un conjunto de ficheros.

Ejemplo.

C:\>ATTRIB +r c:\princip\program*.*

Con esta operación estaremos poniendo el atributo de lectura a todos los ficheros del directorio PROGRAM.

XCOPY (externo)

Copia árboles de archivos y directorios.

SINTAXIS

XCOPY origen [destino] [/A | /M] [/D[:fecha]] [/P] [/S [/E]] [/V] [/W]

[/C] [/I] [/Q] [/F] [/L] [/G] [/H] [/R] [/T] [/U]

[/K] [/N] [/O] [/X] [/Y] [/-Y] [/Z]

[/EXCLUDE:archivo1[+archivo2][+archivo3]...]

origen Especifica el archivo(s) que hay que copiar.

destino Especifica la ubicación y/o nombres de archivos nuevos.

/A Copia solo archivos con el atributo del archivo establecido, no cambia el atributo.

/M Copia solo archivos con el atributo del archivo establecido, desactiva los atributos del archivo.

/D:m-d-y Copia archivos cambiados durante o después de la fecha. Si no se da una fecha, copia solo aquellos archivos cuya fecha de origen es más nueva que la fecha de origen.

/EXCLUDE:archivo1[+archivo2][+archivo3]... Especifica una lista de archivos que contienen cadenas. Los archivos deben tener una cadena por línea. Cuando cualquiera de las cadenas que coincidan con cualquier parte de la ruta absoluta del archivo que se debe copiar, ese archivo se excluirá de ser copiado. Por ejemplo, el especificar una cadena como \obj\ u .obj excluirá todos los archivos por debajo del directorio obj o todos los archivos con la extensión obj respectivamente.

/P Pregunta antes de crear cada archivo de destino.

/S Copia directorios y subdirectorios ,exceptuando los vacíos.

/E Copia directorios y subdirectorios, incluyendo los vacíos. Igual que /S /E. Puede ser usado para modificar /T.

/V Verifica el tamaño de cada archivo nuevo.

/W Pide que se presione una tecla antes de copiar.

/C Continua copiando incluso si ocurren errores.

/I Si el destino no existe y hay que copiar más de un archivo, se asume que el destino debe ser un directorio.

/Q No muestra nombres de archivos mientras copia.

/F Muestra todo el origen y el destino de los nombres de archivo mientras se copian.

/L Muestra los archivos que serán copiados.

/G Permite la copia de archivos destino cifrados para el destino que no es compatible con el cifrado.

/H Copia archivos ocultos y también archivos del sistema.

/R Sobrescribe archivos de solo lectura.

/T Crea una estructura de directorio, pero no copia archivos. No incluye directorios o subdirectorios vacíos. /T /E incluye directorios y subdirectorios vacíos.

/U Copia solo archivos que ya existen en el destino.

/K Copia atributos. Xcopy normal, restablecerá atributos de solo lectura.

/N Copia utilizando los nombres cortos generados.

/O Copia archivos de propiedad e información ACL.

/X Copia archivos de configuración de auditoría (implica /O).

/Y Suprime la pregunta de confirmación para sobrescribir un archivo de destino existente.

/-Y Pregunta por la confirmación de sobrescritura de un archivo de destino existente.

/Z Copia archivos de red en modo reiniciable.

El modificador **/Y** debe ser prestablecido en la variable de entorno **COPYCMD.**

Esto se puede sobrescribir con **/-Y** en la línea del comando.

Esta orden tiene un funcionamiento parecido a la orden **COPY,** con la diferencia de que además de copiar uno o varios archivos, permite copiar estructuras completas de directorios.

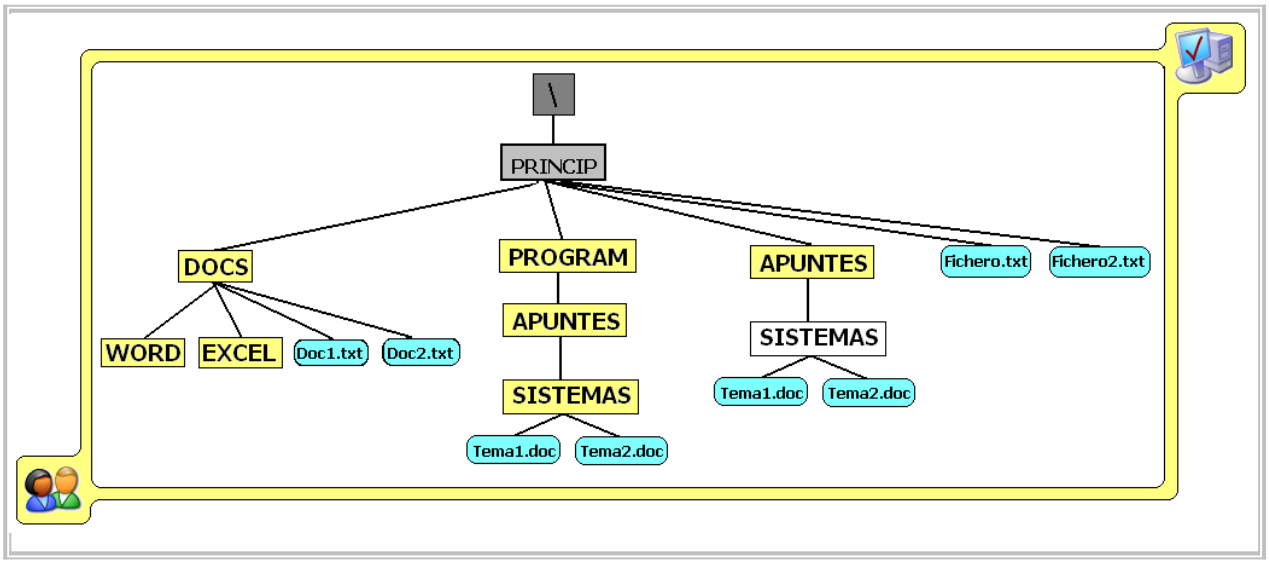
Ejemplo.

Recordemos la estructura de ficheros y directorio con la que hemos trabajado en toda la unidad (figura 1)

Supongamos que queremos copiar sobre el directorio PROGRAM toda la estructura de ficheros y directorios que cuelgan de APUNTES. La orden es, suponiendo que estamos en el directorio raíz, la siguiente:

C:\>XCOPY c:\princip\apuntes c:\princip\program /e/s

El resultado de la copia sería el siguiente:



Para el manejo correcto de esta orden, las consideraciones son las mismas que para la orden **COPY.** Es necesario:

- Saber dónde estamos.
- Saber cuál es el origen de los datos.
- Saber cuál es el destino de los datos.
- Cuáles son los parámetros para realizar la copia deseada.

Con esta orden hay que tener cuidado al copiar estructuras de directorios, ya que no se puede copiar una estructura de ficheros y directorios en otro lugar en el que esté contenido algún directorio que sea el origen de la estructura que vamos a copiar.

Ejemplo.

La siguiente orden no es correcta:

C:\>XCOPY c:\ c:\princip /s/e

Pues queremos copiar el directorio raíz sobre un directorio que depende de él, que además formaría parte del origen de datos. El destino no puede contener el origen si éste es el mismo o parte de él.

8.12. Manejo de trayectorias.

http://commandwindows.com/command1.htm

PATH *(interno)*

Muestra o establece una ruta de búsqueda para archivos ejecutables

SINTAXIS

PATH [[unidad:]ruta[;...][;%PATH%]

PATH ;

Escriba **PATH;** para borrar todas las rutas de búsqueda establecidas y hacer que cmd.exe busque únicamente en el directorio actual. Al incluir **%PATH%** en la nueva ruta de acceso hará que la ruta de acceso anterior se anexe a la nueva configuración.

Es un comando que indica al sistema dónde debe buscar archivos ejecutables (archivos provistos de la extensión COM, EXE o BAT) que no se encuentran en el directorio activo.

Las trayectorias de búsqueda son los caminos que ha de seguir el sistema para localizar los archivos de comandos. Pueden introducirse varias rutas, en cuyo caso deben ir separadas por punto y coma (;).

Ejemplo.

PATH=c:\;c:\princip;c:\dos; etc.

Esta orden es importante, sobre todo cuando queremos ejecutar ficheros. Cuando introducimos el nombre de un fichero ejecutable u orden ejecutable, Windows procede de la siguiente forma: primero, busca en el directorio activo el programa u orden para ejecutar. Si no se encuentra, buscará secuencialmente en las trayectorias especificadas en la orden **PATH.**

Como trayectorias, se pueden incluir unidades lógicas diferentes.

PATH= c:\;a:\;c:\dos;a:\fich; etc.

Su utilidad es la misma. Cuando queremos ejecutar un programa se buscará primero en el directorio activo de la unidad activa. A continuación, en el directorio raíz de C:; seguidamente, en el directorio raíz de A:, etcétera.

Este orden puede ejecutarse directamente tras el símbolo del sistema, o incluirse de forma permanente en el fichero **AUTOEXEC.BAT**, del que hablaremos más adelante. La diferencia es que si no lo incluimos en el fichero **AUTOEXEC.BAT**, tendremos que teclearla siempre que inicialicemos el ordenador. Al incluirla en este fichero, se ejecutará de forma automática cada vez que se encienda el equipo.

Otras funciones de esta orden son las siguientes:

- **PATH** sirve para conocer el **PATH** activo en cualquier momento.
- **PATH**; elimina las rutas de búsqueda establecidas.

APPEND *(interno)*

Permite que los programas abran archivos que se encuentren en los directorios especificados como si estuviesen en el directorio actual.

SINTAXIS

APPEND [[unidad:]ruta[;...]] [/X[:ON | :OFF]] [/PATH:ON | /PATH:OFF] [/E]

APPEND ;

[unidad:]ruta Especifica una unidad y un directorio al que aplicar **APPEND.**

/X:ON Aplica los directorios especificados a las búsquedas de archivos y la ejecución de aplicaciones.

/X:OFF Aplica los directorios especificados solo a las peticiones de apertura de archivo. **/X:OFF** es el valor predeterminado.

/PATH:ON Aplica los directorios añadidos a las peticiones de archivos que ya especifican una ruta. **/PATH:ON** es el predeterminado.

/PATH:OFF Desactiva el efecto de **/PATH:ON.**

/E Guarda una copia de la lista de directorios añadidos en una variable de entorno llamada **APPEND. /E** solo puede ser usado la primera vez que utilice **APPEND** después de iniciar su sistema.

Escriba **APPEND ;** para borrar la lista de directorios añadidos.

Escriba **APPEND** sin parámetros para visualizar la lista de directorios añadidos.

Selecciona un camino de búsqueda para los ficheros de datos, que son aquéllos que no tienen extensión COM, EXE ni BAT. Si lo tecleamos solo, visualizamos las rutas de búsqueda que tiene establecidas.

Las trayectorias de búsqueda son los caminos que ha de seguir el sistema para buscar los archivos de datos. Como en el comando anterior, si se introduce más de una ruta, éstas van separadas por punto y coma (;).

Tiene la misma función que la orden **PATH,** pero sirve para localizar ficheros que no sean ejecutables.

8.13. Redireccionamiento.

Los ficheros que maneja el sistema operativo pueden ser de entrada o de salida. Ya conocemos los diferentes periféricos de entrada o de salida que están que están asociados al manejo de ficheros.

Si necesitamos crear un fichero, es decir, introducir datos en un fichero, tendremos que utilizar los periféricos de entrada necesarios. Si lo que queremos es visualizar o imprimir un fichero, tendremos que utilizar los correspondientes periféricos de salida. Por defecto, el periférico de entrada es el teclado y el periférico de salida el monitor.

Genéricamente, el sistema operativo trabaja con cinco dispositivos diferentes. La gestión de estos dispositivos está incluida dentro de los ficheros que conforman el núcleo del sistema operativo

Los dispositivos que el sistema operativo y otros sistemas operativos reconocen por sí solos son:

- **CON.** Teclado y monitor.
- **PRN, LPT1, LPT2 y LPT3.** Puertos paralelo en los que normalmente hay conectadas impresoras.
- **AUX, COM1, COM2, COM3 y COM4.** Puertos serie en los que hay conectado cualquier tipo de dispositivo periférico.
- **NUL.** Dispositivo nulo.
- **CLOCK&.** Reloj.

Los dispositivos CON son unidireccionales, ya que transfieren la información en un único sentido. El teclado es un dispositivo unidireccional de entrada, pues sirve para introducir datos, y el monitor es un periférico unidireccional de salida, ya que sirve para visualizar datos. En la actualidad existen monitores bidireccionales que sirven para visualizar y para introducir datos.

Los dispositivos LPT1 son bidireccionales, ya que permiten enviar y recibir datos. Se llaman puertos paralelos, centronics o puertos ECP, SPP, EPP, etc. Transmiten información en bloques de 8 bits; es decir, byte a byte. Físicamente son conectores de 25 pines hembra.

El puerto LPT1 se suele llamar PRN y es el primero de ellos. Normalmente, los ordenadores solamente incorporan este puerto paralelo. Por eso es por lo que da igual referenciarlo como LPT1 o como PRN. Si existe alguno más, se llamará LPT2 o LPT3, pero no PRN2 ni PRN3.

Los puertos COMn son bidireccionales y transmiten información bit a bit. Los 8 bits que componen un byte se envían secuencialmente de uno en uno. Se denominan RS-232. Son conectores de 9 o 25 pines macho.

El primer puerto serie, COM1, también se denomina AUX. El resto se denominará COM2, COM3 y COM4.

El dispositivo NUL se utiliza cuando el usuario quiere enviar información a un sitio sin que quede reflejo de ella. También sirve para simular una entrada ficticia.

El dispositivo CLOCK$ sirve para retocar, si fuese necesario, la hora y fecha del sistema.

SÍMBOLO	DESCRIPCIÓN	SINTAXIS
<	Redirección de entrada	Orden<fichero/dispositivo
>	Redirección de salida	Orden>fichero/dispositivo
>>	Adición	Orden>>fichero

Cada dispositivo lleva asociado un nombre que no es modificable y que no puede ser tomado por ningún nombre de archivo ni directorio.

En algunas ocasiones puede ser interesante o necesario cambiar la entrada o salida por defecto. Para ello, lo que haremos será **redireccionar** otro dispositivo de entrada o salida. Los redireccionamientos utilizados son los mostrados en siguientes esquema:

Para ver un primer ejemplo, basta con teclear la orden **DIR** a continuación del símbolo del sistema. De esta forma veremos que se nos muestra por la salida estándar la estructura de archivos y directorios que contiene nuestro disco. Es decir, hemos utilizado la entrada estándar (teclado) y la salida estándar (monitor).

A. Redirección de entrada `<`

La redirección de entrada consiste en cambiar a otro dispositivo o fichero el dispositivo de entrada estándar. Si nosotros ejecutamos la orden **DATE,** el sistema operativo esperará a que el usuario introduzca por la entrada estándar la nueva fecha.

Ejemplo.
Si quisiéramos que el sistema tomase siempre la misma fecha, estaríamos obligados a introducir, cada vez que cargásemos el sistema operativo, la nueva fecha; no obstante, hay una alternativa: redireccionar la entrada estándar. Si nosotros grabamos en un fichero la fecha deseada (15/12/97) y lo redireccionamos como entrada hacia la orden **DATE,** lo que estaremos consiguiendo es no tener que teclear la

fecha, sino que la orden tome como datos de entrada los del fichero en el que hemos grabado la fecha. Supongamos que este fichero, en el que hemos introducido la fecha con el formato adecuado, se llama fecha.txt. La orden para que el sistema siempre tomase esa fecha como la del sistema sería la siguiente:

C:\DATE<fecha.txt

Hemos redireccionado la entrada estándar del teclado por un fichero.

El sistema operativo cuenta, además, con tres ordenes especialmente diseñadas para ser utilizadas por el redireccionamiento de entrada. Son las siguientes:

- **SORT.** Permite ordenar los datos entrada.
- **MORE.** Visualiza el contenido de los datos de entrada fraccionando su presentación en pantallas completas.
- **FIND**. Localiza una cadena de caracteres concretos dentro de los datos de entrada.

B. Redireccionamiento de salida `>`

Este redireccionamiento es mucho más utilizado que el de entrada. Especialmente se utiliza para dos operaciones: enviar la salida de un programa a la impresora o enviar la salida de un programa a un fichero.

Si lo que queremos es redireccionar la salida de un programa u orden concreta a un dispositivo diferente a la salida estándar (por ejemplo, la impresora), bastará con poner la orden concreta y, a continuación, **>LPT1** o **PRN.** De esta forma, lo que estamos consiguiendo es que la información se imprima y que no se visualice. La información que se nos mostrará en pantalla saldrá directamente por la impresora.

Si lo que deseamos es que el resultado de la ejecución de una orden o programa concreto se almacene en un fichero, bastará con ejecutar la orden o programa deseado y, a continuación, redireccionar la salida a un fichero. Para ello, bastará con poner la orden y, a continuación, **>nombre_fichero.**

Ejemplo.

Un ejemplo claro del redireccionamiento de salida es el de enviar a la impresora el resultado de la orden **DIR.** Si nosotros ponemos:

C:\>DIR>PRN

O bien:

C:\>DIR>LPT1

Veremos que el resultado de la orden **DIR** sale por la impresora.

El mismo resultado obtendríamos ejecutando la siguiente secuencia de órdenes:

C:\>DIR>salida

C:\>TYPE salida>PRN

En este ejemplo, lo que hacemos es redireccionar la salida estándar a un fichero llamado salida. Posteriormente, ejecutamos la orden para visualizarlo, pero redireccionándolo hacia la impresora. En definitiva, el resulta de la orden **DIR** sale por la impresora.

Si, como en el último ejemplo, el fichero salida ya existía, la información de éste de destruye, sustituyéndose por la nueva información que le enviamos a través del redireccionamiento. Si el fichero salida no existe, se crea.

C. Adición a un fichero `>>`

Con este redireccionamiento siempre se crea un fichero nuevo. Si el fichero no existe, se crea por primera vez. Si el fichero existe, conserva la información que tenía antes y se le añade la nueva información al final del fichero.

Este tipo de redireccionamiento *(append)* se usa exclusivamente para añadir más información a un mismo fichero. Cada vez que se redirecciona con la adición el mismo fichero, se aumentará su contenido.

Ejemplo.

Supongamos que generamos un fichero, desde el teclado, utilizando la orden **COPY.** Llamamos al fichero texto.

C:\>COPY CON texto

Este es el contenido del fichero:

Ctrl+z+Enter

1 Fichero(s) copiado(s)

C:\>

Supongamos ahora que queremos hacer una copia de este fichero en el mismo lugar. El nuevo fichero se llamará copia. Para ello podemos hacer lo siguiente:

C:\>COPY texto copia

O utilizar el redireccionamiento:

C:\>TYPE texto>copia

El resultado es el mismo.

Si a continuación ponemos:

C:>TYPE texto>>copia

Estaremos añadiendo más información al fichero copia. Concretamente, el fichero copia tendría el siguiente contenido:

Este es el contenido del fichero

Este es el contenido del fichero

Si volvemos a ejecutar la orden:

C:\>type texto>>copia

El contenido del fichero sería:

Este es el contenido del fichero

Este es el contenido del fichero

Este es el contenido del fichero

8.14. Edición de textos en entorno comando.

http://commandwindows.com/command1.htm

Para editar ficheros de texto como los propios de configuración del sistema autoexec.bat y config.sys, se utiliza el edit.com, que es un fichero ejecutable del sistema operativo. Este fichero es un editor de textos similar a los editores actuales, pero con muchas menos prestaciones. Su escasa potencia hace que sea muy fácil de utilizar.

El fichero ejecutable del editor de textos es el **edit.com,** pero este fichero en sí no es el propio editor. Este archivo hace referencia a otro, llamado **qbasic.exe,** que es en realidad el fichero ejecutable del editor de textos.

EDIT *(externo)*

Editor de texto plano en entorno comando (plataformas de 32 bits).

SINTAXIS

[EDIT [/B] [/H] [/R] [/S] [/<nnn>] [/?] [archivos]

/B - Fuerza a modo monocromo.

/H - Muestra el número máximo de líneas posible para su hardware.

/R - Carga archivos en modo de solo lectura.

/S - Fuerza el uso de nombres de archivos cortos.

/<nnn> - Carga archivos binarios, ajustando líneas a todos los caracteres <nnn>.

/? - Muestra la Ayuda en pantalla.

[archivo] - Especifica qué archivos iniciales cargar. Se pueden usar comodines y especificaciones múltiples.

En cualquier caso, para editar un texto ejecutaremos el editor de textos de la siguiente manera:

C:\>EDIT

Si empezamos el trabajo de esta forma, se supone que vamos a crear un nuevo fichero. El fichero, *a priori*, no tiene ningún nombre. En la parte central superior aparece sin título. Antes de abandonar el editor tendremos que grabarlo asignándole un nombre. Para ello, utilizaremos las opciones **Guardar** o **Guardar como.**

Esta misma operación la podemos hacer poniendo lo siguiente:

C:\>EDIT texto

De esta forma, si el fichero texto existe, se edita para que lo podamos modificar. Si no existe, se crea.

Este editor de texto funciona de forma similar a cualquier otro, si bien en entorno texto. Funciones como cortar, copiar, pegar, imprimir, salir, nuevo, etc., son las misma de cualquier editor de textos. En este no se pueden insertar gráficos, solamente caracteres ASCII convencionales.

Para editar un fichero concreto no es necesario que estemos en el directorio en el que se encuentra. Tampoco es necesario estar en el directorio en el que se encuentra el programa edit.com.

Aplicaremos lo aprendido en la unidad anterior sobre las trayectorias absolutas y relativas y tendremos en cuenta las órdenes **APPEND** y **PATH** como trayectorias en las que se encuentran archivos y programas, respectivamente.

Para activar las opciones de la barra de menús basta con pulsar la tecla **Alt** y luego utilizar las flechas de edición de derecha o izquierda para movernos por las diferentes opciones, y las flechas de arriba y abajo para desplegar las opciones de cada opinión principal. Para movernos por otras partes del editor, utilizaremos la tecla **Tab** (tecla con dos flechas a izquierda y derecha situada en la parte izquierda del teclado).

8.15. Ficheros BATCH o por lotes en Windows.

http://commandwindows.com/command1.htm

Se dice que un fichero es de tipo BAT (BATCH) o de tratamiento por lotes cuando está formado por un grupo o lista de órdenes del sistema operativo que se ejecutan una tras otra.

En determinados casos, este tipo de ficheros nos permite ahorrar tiempo, ya que, una vez grabado en disco, con solo teclear su nombre se ejecutan las órdenes que forman la lista.

Para crear un fichero BAT, se utiliza el un editor de texto plano.

Ejemplo.
Un ejemplo podría ser el fichero llamado primero.bat, compuesto por las órdenes: **DATE** **CLS** **DIR** **VER** Para ser ejecutado, basta con poner su nombre tras el símbolo del sistema. Recordemos las trayectorias: un fichero, para su ejecución, no tiene que estar necesariamente en el directorio activo. Puede estar en cualquier sitio y ejecutarse indicando su trayectoria absoluta, relativa, o bien incluyendo en la orden **PATH** la ruta. Si ponemos: **C:\>primero** Se ejecutarán las cuatro órdenes anteriores.

Entre las características de este tipo de ficheros podemos destacar las siguientes:

- Todos llevan extensión BAT.
- Cada orden ocupa una línea de la lista.
- Podemos incluir órdenes internas y externas del sistema operativo, comandos especiales de los ficheros BAT, e incluso pueden llamar a otros ficheros ejecutables, que pueden ser otros BAT.
- No se puede asignar como nombre de un fichero BAT el de una orden externa o interna, ya que entonces solo se ejecutará ésta y no el fichero.
- Podemos interrumpir la ejecución de un fichero BAT pulsando simultáneamente **Control+C** o **Control**+**BREAK.**

- Una vez guardado en disco, solo es preciso teclear su nombre, sin indicar extensión, para que se ejecute.

Si en un directorio concreto existe un fichero llamado program.exe y otro llamado program.bat, siempre se ejecutará primero el de extensión BAT. Es decir, si queremos ejecutar cualquiera de ellos, tendremos que especificar además su extensión.

A. Comandos de ficheros BATCH.

A parte de todos los comandos propios del sistema operativo, un fichero BATCH puede contener una serie de comandos especiales de control mediante los cuales podemos controlar el flujo de ejecución del comando, así como poner comentarios, instrucciones condicionales, etc. Estos comandos se llaman **comandos de procesamiento por lotes.**

ECHO

Este comando puede ser utilizado para dos propósitos:

- Activar o desactivar la visualización de los comandos en la pantalla.
- Mostrar un mensaje en pantalla.

SINTAXIS

ECHO [ON OFF] [MENSAJE]

ECHO. Muestra en pantalla el estado actual de eco.

ECHO ON. Activa el eco y se ven en pantalla todos los mensajes.

ECHO OFF. Desactiva el eco y no se ven en pantalla los mensajes hasta que se vuelva a activar.

ECHO *mensaje*. Visualiza en pantalla el mensaje que deseamos, independientemente de que el eco esté o no activado.

Normalmente, los comandos incluidos dentro de un fichero BATCH se visualizan en la pantalla a medida que se van ejecutando; esto es, tienen un eco. Si esto ocurre el **ECHO** estarán ON.

Si queremos que los comandos no aparezcan en la pantalla, podemos desactiva el **ECHO** con la opción OFF.

Normalmente, se pone en la primera línea de programa la orden **ECHO OFF** si no queremos ver ningún mensaje de llamada. Lo que sí aparecerá será la ejecución de la propia orden, ya que para sí misma aún sigue activa.

Si queremos que no se vea ni siquiera la propia orden que desactiva el eco, colocaremos el carácter **@** delante de **ECHO OFF.** De esta forma no veremos nada..

REM

Permite incluir comentarios en los ficheros por lotes.

SINTAXIS

REM [COMENTARIO]

Permite incluir comentarios explicativos en un archivo de proceso por lotes. Este mensaje

será visualizado solo en el caso de que esté activado el comando de procesamiento por lotes **ECHO.** Podemos usar **REM** sin comentarios para añadir espacios que faciliten la lectura.

Esta orden no ejecuta nada, simplemente introduce el comentario que, posteriormente, puede ser visualizado o no. Es recomendable incluir comentarios en los ficheros de procesos por lotes, con el fin de clarificar qué es lo que hacemos con cada orden o conjunto de órdenes. Ningún mensaje de llamada. Lo que sí aparecerá será la ejecución de la propia orden, ya que para sí misma aún sigue activa.

Si queremos que no se vea ni siquiera la propia orden que desactiva el eco, colocaremos el carácter **@** delante de **ECHO OFF.** De esta forma no veremos nada.

PAUSE

Pause la ejecución de un fichero por lotes.

SINTAXIS

PAUSE [COMENTARIO]

Cuando se está ejecutando un archivo de procesamiento por lotes, puede ser necesario cambiar de disco o realizar alguna acción. Este comando suspende la ejecución del archivo hasta que se presione cualquier tecla, a menos que se presione **Control+BREAK** o **Control+C**, en cuyo caso se suspenderá la ejecución del comando.

Este parámetro es útil cuando se desea mostrar un mensaje especial.

Suspende temporalmente la ejecución de un archivo de procesamiento por lotes

B. Ficheros BATCH con parámetros.

Pueden darse ocasiones en las que se quiera crear un programa y ejecutarlo con diferentes conjuntos de datos. Estos datos pueden ser almacenados en varios archivos de Windows. Estos parámetros se introducen desde el símbolo del sistema, tras el nombre del fichero de procesamiento por lotes.

Centro de un fichero BATCH es posible emplear hasta 9 parámetros, los cuales se identifican con los caracteres **%n**, donde **n** es un número comprendido entre 1 y 9. Es decir, dentro de un fichero podemos emplear los siguientes parámetros: **%1, %2**.....**%9.**

Si nuestro fichero BATCH incluye un parámetro, este será sustituido por el nombre introducido tras el nombre del fichero por lotes.

Ejemplo.

Supongamos que queremos realizar un programa BATCH que visualice, imprima y copia un fichero concreto al directorio raíz de la unidad A:. Mediante el editor de textos, crearíamos un programa ejemplo.bat con el siguiente contenido:

@ECHO OFF

TYPE %1

PRINT %1

COPY %1 a:

Una vez creado el programa, si lo ejecutamos de la siguiente forma:

C:\>ejemplo

lo único que se produciría sería un error. El siguiente:

Falta el parámetro requerido.

La ejecución correcta sería la siguiente:

C:\>ejemplo fichero.txt

Es evidente que fichero.txt es el parámetro que estamos pasando al programa por lotes.

Cada parámetro es posicional respecto de lo que introducimos tras el nombre del fichero de proceso por lotes. Si introducimos dos parámetros, **%1** y **%2,** el fichero BATCH esperará a que se le introduzcan dos parámetros tras el nombre del fichero. Cada parámetro se introduce separado por un espacio en blanco. El primer parámetro se sustituirá por **%1;** el segundo, por **%2,** y así sucesivamente, pero de forma fija en todo el proceso.

Ejemplo.

Supongamos que creamos el fichero ejemplo.bat para que visualice el primer fichero que introduzcamos, imprima el segundo y copie el tercero:

@ECHO OFF

TYPE %1

PRINT %2

COPY %3 a:

Para ejecutar el fichero, pondríamos:

C:\>ejemplo texto1.txt fich2.txt program.exe

De esta forma, texto1.txt se visualiza, texto2.txt se imprime y en el directorio raíz de la unidad A: copiamos el fichero program.exe.

Si en este ejemplo introducimos más de tres parámetros, los restantes serán ignorados. Si introducimos menos, se producirá un error indicando que falta algún parámetro.

La ventaja de trabajar con parámetros reemplazables es que un mismo fichero BATCH puede ser utilizado para cualquier fichero de nuestro ordenador, siendo el nombre del fichero el parámetro pasado al fichero de proceso por lotes. Si no existiesen los parámetros, tendríamos que repetir el fichero BATCH para cada archivo que lo utilizase.

C. Instrucciones BATCH para control del proceso de ejecución.

IF

Realiza el procesamiento condicional de los programas por lotes.

SINTAXIS

IF [NOT] ERRORLEVEL número comando

IF [NOT] cadena1==cadena2 comando

IF [NOT] EXIST archivo comando

NOT Especifica que Windows XP debe llevar a cabo el comando solo si la condición es falsa.

ERRORLEVEL número Especifica una condición verdadera si el último programa que se ejecutó devolvió un código de salida igual o mayor que el número especificado.

cadena1==cadena2 Especifica una condición verdadera si las cadenas de texto especificadas coinciden.

EXIST archivo Especifica una condición verdadera si el archivo Especificado existe.

comando Especifica el comando que se ejecutará si se cumple la condición. Comando

puede ir seguido de la palabra. clave **ELSE** y, a continuación, un comando que se ejecutará si la condición especificada es FALSA. La cláusula **ELSE** debe aparecer en la misma línea que la del comando que sigue al **IF** Por ejemplo:

```
IF EXIST archivo. (
del archivo.
) ELSE (
echo archivo. no existente.
)
```

Lo siguiente NO funcionará porque el comando **DEL** debe terminarse con una nueva línea:

```
IF EXIST archivo. del archivo. ELSE echo archivo. no existente
```

Tampoco funcionará lo siguiente, ya que el comando ELSE debe estar en la misma línea del comando IF:

```
IF EXIST archivo. del archivo.
ELSE echo archivo. no existente
```

Si desea tenerlo todo en una misma línea, lo siguiente funcionaría:

```
IF EXIST archivo. (del archivo.) ELSE echo archivo. no existente
```

Si los comandos de extensión están habilitados IF cambia así:

```
IF [/I] cadena1 op-de-comparación cadena2 comando
IF CMDEXTVERSION número comando
IF DEFINED variable comando
```

donde ***op-de-comparación*** puede ser:

- EQU – igual.
- NEQ - no igual.
- LSS - menor que.
- LEQ - menor que o igual.
- GTR - mayor que.
- GEQ - mayor que o igual.

y el modificador **/I,** si se especifica, realiza comparaciones de cadena que no distinguen entre mayúsculas y minúsculas. El modificador **/I** también puede usarse en la forma cadena1==cadena2 de IF. Estas comparaciones son genéricas, por lo que si tanto cadena1 como cadena2 se constituyen únicamente por dígitos numéricos, entonces las cadenas se convierten a números y se realiza una comparación numérica.

El condicional CMDEXTVERSION funciona solo como ERRORLEVEL, excepto si se compara con un número de versión interna asociada con las extensiones de comando.

La primera versión es 1. Será incrementada en uno cuando las significantes mejoras sean agregadas a las extensiones de comando.

El condicional CMDEXTVERSION nunca es verdadero cuando las extensiones de comando están deshabilitadas.

El condicional DEFINED funciona solo como EXISTS excepto cuando toma un nombre de

variable de entorno y vuelve como verdadero si se define la variable de entorno.

%ERRORLEVEL% se expandirá a una representación de cadena del valor actual de ERRORLEVEL, siempre y cuando no exista ya una variable de entorno con el nombre ERRORLEVEL, en cuyo caso obtendrá su valor.

Después de ejecutar un programa, lo siguiente ilustrará el uso de ERRORLEVEL.

```
goto answer%ERRORLEVEL%
:respuesta0
echo Programa ha devuelto el código 0
:respuesta1
echo Programa ha devuelto el código 1
```

También puede usar las comparaciones numéricas anteriores:

```
IF %ERRORLEVEL% LEQ 1 goto okay
```

%CMDCMDLINE% se expandirá a una línea de comando original pasada al anterior CMD.EXE a cualquier proceso CMD.EXE, siempre y cuando no exista ya una variable de entorno con el nombre CMDCMDLINE, en cuyo caso obtendrá su valor.

%CMDEXTVERSION% se expandirá a una representación de la cadena del valor actual CMDEXTVERSION, siempre y cuando no exista ya una variable de entorno con el nombre CMDEXTVERSION, en cuyo caso obtendrá su valor.

Ejemplo.

Supongamos que queremos realizar un programa BATCH, llamado formateo.bat, que permita dar formato a disquetes que tengan un tamaño de 720 o de 1 440 Bytes. Si el formato es correcto indicaremos por pantalla tal acción o, en caso contrario, que no ha sido correcta. Para ello, vamos a introducir dos parámetros: unidad lógica de disquete (A: o B:) y tamaño del disquete:

```
@echo off
rem Programa para formatear disquetes
format %1 /f:%2
IF ERRORLEVEL 0 ECHO Formato realizado con éxito
IF NOT ERRORLEVEL 0 ECHO Formato NO realizado
```

En este caso, al ejecutar el programa tendríamos que proceder de la siguiente forma:

```
C:\>Formateo A: 1.44
```

%1 toma como valor A:, y %2 toma como valor 1.44. Si todo es correcto, se muestra el mensaje adecuado.

Ejemplo.

Supongamos que queremos saber si se ha introducido el nombre de la unidad que vamos a formatear. La orden sería la siguiente:

IF “%1”==““ echo ERROR EL PARAMETRO NO TIENE VALOR

Si no se ha introducido el parámetro, su valor se sustituye por espacios en blanco. Como lo que se compara es la primera cadena (que son “espacios en blanco”) con la segunda cadena (que también son “espacios en blanco”, ya que no hemos puesto nada), la condición sería correcta.

En este caso, enviaríamos el correspondiente mensaje de error e interrumpiríamos la ejecución del proceso BATCH.

Ejemplo.

Supongamos que queremos saber si existe un fichero llamado fichero.txt en el directorio raíz de la unidad A:. Pondremos lo siguiente:

IF EXIST a:\fichero.txt ECHO El fichero si existe

Si queremos saber si existe, por ejemplo, el directorio datos como directorio hijo del directorio raíz de la unidad A:, tendríamos que poner lo siguiente:

IF EXIST a:\datos\nul ECHO El directorio si existe

En este caso, comprueba la existencia del dispositivo NUL dentro del directorio datos. Como este fichero (dispositivo) siempre existe, no se produce error. Si existe realmente el directorio datos, aunque no tenga ningún fichero o no tenga el que nosotros deseamos, sabremos si existe o no.

GOTO

Dirige cmd.exe a una línea con etiqueta en un programa por lotes

SINTAXIS

GOTO etiqueta

etiqueta Especifica el texto usado como etiqueta en el programa.

Debe escribir una etiqueta en una línea, empezando con dos puntos (:).

Si las extensiones de comando están habilitadas, **GOTO** cambia así:

El comando **GOTO** acepta ahora una etiqueta de destino: **EOF** que transfiere control al final del archivo de script por lotes actual. Ésta es una manera fácil de salir de un archivo de script por lotes de script sin tener que definir una etiqueta. Escriba **CALL /?** para obtener una descripción de las extensiones del comando **CALL** que hace útil esta característica.

Ejemplo.

Supongamos, a continuación, el proceso formateo.bat. Vamos a modificar algunas condiciones, de tal forma que si el formato es correcto se visualice el mensaje que ha sido correcto, y además vamos a copiar el archivo datos.dat del directorio raíz de la unidad C: en el directorio raíz de la unidad recién formateada con el nombre nuevo.dat. En caso de error no haremos nada.

El programa formateo.bat sería el siguiente:

```
@ ECHO OFF
REM programa para formatear disquetes
FORMAT %1 /f:%2
IF NOT ERRORLEVEL 0 GOTO :INCORRECTO
IF ERRORLEVEL 0 GOTO :CORRECTO
:CORRECTO
ECHO El disco se ha formateado con éxito
COPY C:\datos.dat a:\nuevo.dat
GOTO :FIN
:INCORRECTO
ECHO El disco no se ha formateado
:FIN
```

CHOICE (Hasta Windows 7)

Dirige cmd.exe a una línea con etiqueta en un programa por lotes.

SINTAXIS

CHOICE TEXTO] [/c[:TECLAS][/n][/s][/t][:]TECLA,SEGUNDOS]

Esta orden cumple una triple función en los ficheros BATCH:

- Muestra un mensaje en pantalla y detiene la ejecución por lotes.
- Espera a que el usuario pulse una tecla previamente prediseñada.
- Activa un código de salida *ERRORLEVEL* según la tecla pulsada.

Se utiliza principalmente para permitir el usuario introducir parámetros en tiempo de ejecución del proceso BATCH.

TEXTO: es lo que queremos que se visualice cuando se ejecute la orden **CHOICE.**

/c:teclas*:** Se utiliza para definir las teclas que el usuario puede pulsar cuando se ejecuta la orden. Si deseamos que esta orden reconozca como teclas válidas correspondientes a los caracteres **S** y **N,** especificaremos: ***CHOICE /c:SN

Cuando el usuario pulsa una de ellas, se activa el código de salida de *ERRORLEVEL*, correspondiente a la posición de la tecla en el parámetro /**C.** El carácter **S** devolvería una salida para ERRORLEVEL de 1. El carácter N devolvería 2. Da igual que sean mayúsculas o minúsculas.

/n visualiza primero en pantalla la cadena de caracteres del parámetro TEXTO, y luego el mensaje de teclas.

/t se utiliza para introducir qué tecla se asumirá por defecto poner como directorio activo sin el usuario no pulsa ninguna pasado un determinado tiempo que se especifica en segundos.

Ejemplo.

Supongamos que queremos realizar un fichero BAT que nos ponga como directorio activo

uno u otro en una unidad concreta, dependiendo de la tecla que pulsemos. El fichero se llamará directo.bat y nos pondrá como directorio y unidad activa el directorio raíz de A: si pulsamos la tecla A, en el raíz de C: si pulsamos C y en c:\dos si pulsamos B. No hay tecla por defecto:

```
@ECHO OFF
CLS
REM Proceso para situarse en un directorio
ECHO
ECHO A. Para poner directorio activo a:\
ECHO B. Para poner directorio activo c:\dos
ECHO C. Para poner directorio activo c:\
ECHO
CHOICE “Pulse una tecla “ /c:ABC
ECHO
IF ERRORLEVEL 3 GOTO :RAIZC
IF ERRORLEVEL 2 GOTO :DOS
IF ERRORLEVEL 1 GOTO :RAIZA
IF ERRORLEVEL 0 GOTO :FIN
:RAIZA
CD A:\
GOTO :FIN
:DOSC
CD C:\DOSC
GOTO :FIN
:RAIZC
CD C:\
:FIN
```

FOR

Ejecuta el comando para cada uno de los archivos especificados en el conjunto de archivos.

SINTAXIS

FOR %variable IN (conjunto) DO comando [parámetros]

FOR %variable IN (conjunto) DO comando [parámetros]

%variable Especifica un parámetro reemplazable de una sola letra.

(conjunto) Especifica un conjunto de uno o más archivos. Se pueden usar comodines.

comando Especifica el comando que se ejecutará para cada archivo.

parámetros Especifica los parámetros o modificadores del comando especificado.

Para usar el comando **FOR** en un programa por lotes, especificar **%%variable** en vez de **%variable.** Los nombres de las variables distinguen entre mayúsculas y minúsculas, por lo tanto **%i** no es lo mismo que %I.

Si las extensiones de comandos están habilitadas, las siguientes formas adicionales del comando **FOR** son compatibles:

FOR /D %variable IN (conjunto) comando DO [parámetros]

Si el conjunto contiene comodines, se especifica para coincidir con el nombre del directorio en vez de los nombres de archivo.

FOR /R [[unidad:]ruta] %variable IN (set) comando DO [parámetros]

Guía el directorio del árbol de raíz a **[unidad:]ruta,** ejecutando la instrucción **FOR** en cada directorio del árbol. Si no se especifica el directorio después de **/R** entonces se asume que es el directorio actual. Si el conjunto es solamente un simple carácter de punto (**.**) entonces enumerará el árbol del directorio.

FOR /L %variable IN (inicio, paso, fin) comando DO [parámetros]

El conjunto es una sucesión de números que va desde inicio hasta fin y que aumenta (o disminuye) en función de lo especificado en paso. Así (1, 1, 5) generará la sucesión 1 12 3 4 5 y (5, -1, 1) generará la sucesión 5 4 3 2 1

FOR /F [“opciones”] %variable IN (conjunto-archivo) comando DO [parámetros]

FOR /F [“opciones”] %variable IN ('cadena') comando DO [comando-parámetros]

FOR /F [“opciones”] %variable IN ('comando')comando DO [comando-parámetros]

o, si la opción **usebackq** está presente:

FOR /F [“opciones”] %variable IN (conjunto-archivo) comando DO [parámetros]

comando DO [comando-parámetros]

FOR /F [“opciones”] %variable IN ('cadena') comando DO [parámetros]

FOR /F [“opciones”] %variable IN (`comando`) comando DO [parámetros]

Conjunto de archivos es uno o más nombres de archivos. Cada archivo es abierto, leído y procesado antes de ir al siguiente archivo en el conjunto de archivos. Procesar consiste en leer el archivo, partirlo en líneas individuales de texto y analizar cada línea en cero o más símbolos. El cuerpo del bucle se llama con los valores de la variable establecidos para las cadenas de símbolo encontradas. De forma predeterminada, **/F** pasa el primer símbolo separado en blanco desde cada línea.

Las líneas en blanco se saltan. Puede sustituir el comportamiento de análisis predeterminado si especifica el parámetro opcional “opciones”. Esto es una cadena entre comillas que contiene una o más palabras claves para especificar diferentes opciones de análisis. Las palabras claves son:

- **eol=c** - especifica un carácter de comentario al final de la línea (solo uno).
- **skip=n** - especifica el número de líneas que hay que saltarse al principio del archivo.
- **delims=xxx** - especifica un grupo de delimitadores. Esto reemplaza al grupo de delimitadores predeterminados de espacio y tabulación.
- **tokens=x,y,m-n** - especifica qué símbolos de cada línea deben pasarse al cuerpo de la cláusula “for” en cada iteración. Esto causará que los nombres de variables

adicionales sean asignados. La forma m-n es un intervalo del símbolo m-ésimo al símbolo n-ésimo. Si el último carácter en la cadena tokens= es un asterisco, se asigna una variable adicional que recibe el resto del texto en la línea posterior al último símbolo analizado.

- **usebackq** - especifica que la nueva semántica está vigente, donde una cadena entre comillas inversas se ejecuta como un comando y una cadena con comillas simples es un comando de cadena literal y permite el uso de comillas dobles para entrecomillar los nombres de archivo en un grupo de nombres de archivos.

Estos ejemplos pueden ayudar:

```
FOR /F “eol=; tokens=2,3* delims=, “ %i in (archivo.txt) do @echo %i %j %k
```

Analizará cada línea en mi archivo.txt excepto las que inicien con un punto y coma, pasando el segundo y tercer símbolo de cada línea al cuerpo de **FOR.**

Los símbolos están delimitados por comas y/o espacios.

Tenga en cuenta que las instrucciones del cuerpo de **FOR** hacen referencia a **%i** para obtener el segundo símbolo, a **%j** para obtener el tercero y a **%k** para obtener el resto de los símbolos posteriores al tercero. Para los nombres de archivo que contengan espacios, necesita poner comillas dobles en los nombres de archivos. Para usar comillas dobles de esta manera también necesita usar la opción **usebackq,** de lo contrario las comillas dobles serán interpretadas para definir el análisis de una cadena literal.

%i está explícitamente declarado en la instrucción y **%i** está explícitamente declarado en la instrucción **FOR, %j** y **%k** están declarados implícitamente a través de la opción **=tokens.** Puede especificar hasta 26 símbolos a través de la línea **=tokens,** siempre y cuando no cause un intento de declarar una variable mayor que la letra 'z' o 'Z'.

Recuerde, los nombres de variables de **FOR** son de una sola letra y distinguen mayúsculas de minúscula. Además, las variables son globales y no puede haber más de 52 variables activas al mismo tiempo. También puede usar la lógica de análisis de **FOR /F** en una cadena inmediata convirtiendo el conjunto de archivos entre paréntesis en una cadena entre comillas usando caracteres con comillas simples. Será tratada y analizada como una simple línea de entrada de un archivo.

Finalmente puede usar el comando **FOR /F** para analizar la salida de un comando.

Se hace convirtiendo el conjunto de archivos entre paréntesis una cadena con comillas invertidas. Se tratará como una línea de comandos que se pasa a un **CMD.**

EXE secundario y la salida es capturada en memoria y evaluada como si fuera un archivo. Como en el siguiente ejemplo:

```
FOR /F “usebackq delims==“ %i IN (`conjunto`) DO @echo
```

%i enumerará los nombres de variable de entorno en el entorno actual.

Además, la sustitución de las referencias de variables **FOR** ha sido mejorada. Ahora puede usar la siguiente sintaxis opcional:

%~I - expande %I quitando las comillas (“) que pudiera haber.

%~fI - expande %I a un nombre de ruta calificado.

%~dI - expande %I solo a una letra de unidad.

%~pI - expande %I solo a una ruta.

%~nI - expande %I solo a un nombre de archivo.

%~xI - expande %I solo a una extensión de archivo.

%~sI - ruta expandida contiene solo nombres cortos.

%~aI - expande %I a atributos de archivos.

%~tI - expande %I a fecha/hora del archivo.

%~zI - expande %I a tamaño del archivo

%~$PATH:I - busca los directorios de la lista de la variable de entorno de **PATH** y expande **%I** al nombre totalmente calificado del primero que se encuentre.

Si el nombre de la variable de entorno no es definido o no se encuentra el archivo en la búsqueda, el modificador se expande a la cadena vacía. Los modificadores se pueden combinar para conseguir resultados compuestos:

%~dpI - expande %I solo a una letra de unidad y ruta.

%~nxI - expande %I solo a un nombre de archivo y extensión.

%~fsI - expande %I solo a un nombre de ruta con nombres cortos.

%~dp$PATH:i - busca los directorios de la lista de la variable de entorno de **PATH** para **%I** y se expande a la letra de unidad y ruta del primero que encuentre.

%~ftzaI - expande **%I** a **DIR** como línea de salida.

En los ejemplos anteriores **%I** y **PATH** pueden ser reemplazados por otros valores válidos. La sintaxis **%~** está terminada por un nombre de variable **FOR** válido. El código se vuelve más legible si se usan variables en mayúscula como **%I.** Además esto evita confundir las variables con los modificadores, los cuales no distinguen entre mayúsculas y minúsculas.

Ejemplo.

Supongamos que queremos realizar un programa BAT, llamado imprimir.bat que visualice tres archivos, denominados doc1.txt, doc2.txt y doc3.txt. Para ello introduciremos dentro del fichero imprimir.bat lo siguiente:

FOR %%V IN (doc1.txt doc2.txt doc3.txt)

DO TYPE %%V

Este mismo ejemplo se podría haber realizado con archivos que no sean específicos. Es decir, los nombres de archivo especificados dentro del conjunto pueden ser aquellos que cumplan una determinada plantilla (por ejemplo, *.txt), o parámetros determinados capaces de sustituir un archivo introducido por teclado antes de ejecutar el programa en cuestión.

Este mismo ejemplo lo podemos hacer de la siguiente forma:

FOR % % V IN (*.TXT) DO TYPE % % V

También podemos realizar un proceso con parámetros, que posteriormente introduciremos desde teclado. La orden sería la siguiente:

FOR % % V IN (%1 %2 %3) DO TYPE % %V

La única diferencia se da a la hora de ejecutar el programa, ya que en los dos casos anteriores pondríamos solamente:

C:\>imprimir

En el último caso tendremos que poner:

C:\>imprimir doc1.txt doc2.txt doc3.txt

Si el número de parámetros introducidos es mayor o menor a los especificados en el conjunto, ocurrirá lo mismo que en cualquier otro proceso que utilice parámetros.

CALL

Llama a un archivo por lotes desde otro.

SINTAXIS

CALL [unidad:][ruta]archivo [parámetros]

parámetros Especifica cualquier información de la lista de comandos que necesita el programa por lotes.

Si las extensiones de comando están habilitadas **CALL** cambia así:

El comando **CALL** acepta ahora etiquetas como el destino de **CALL.**

La sintaxis es:

CALL : argumentos de etiqueta

Se ha creado un nuevo contexto de archivo por lotes con los argumentos especificados y el control se pasa a la instrucción después de especificar la etiqueta. Debe "salir" dos veces al pasar dos veces al final del archivo de comandos por lotes. La primera vez que lea el final, el control volverá a la instrucción que sigue a **CALL.** La segunda vez se cerrará el archivo de comandos por lotes. Escriba **GOTO /?** para obtener una descripción de **GOTO:** la extensión **EOF** le permitirá "regresar" del archivo de proceso por lotes.

Además, la expansión de las referencias de argumento del archivo de comandos por lotes (%0, %1, etc.) se ha cambiado así:

%* en un comando de secuencias por lotes se refiere a todos los argumentos (por ej., %1 %2 %3 %4 %5 ...)

La sustitución de los parámetros del archivo por lotes (%n) ha sido ampliada. Ahora puede usar la siguiente sintaxis opcional:

%~1 - expande **%1** quitando las comillas (") que pudiera haber.

%~f1 - expande **%1** a un nombre completo de ruta.

%~d1 - expande **%1** solo a una letra de unidad.

%~p1 - expande **%1** solo a una ruta.

%~n1 - expande **%1** solo a un nombre de archivo.

%~x1 - expande **%1** solo a una extensión de archivo.

%~s1 - expande la ruta que contiene solo nombres cortos.

%~a1 - expande **%1** a los atributos del archivo.

%~t1 - expande **%1** a la fecha/hora del archivo.

%~z1 - expande **%1** al tamaño del archivo.

%~$PATH: 1 - busca los directorios en la lista de la **RUTA** de la variable de entorno y expande **%1** al nombre totalmente calificado del primero que encontró. Si el nombre de la variable de entorno no se ha definido o no se ha encontrado el archivo en la búsqueda,

entonces este modificador se transforma en una cadena vacía.

Los modificadores se pueden combinar para obtener resultados compuestos:

%~dp1 - expande **%1** solo a una letra de unidad y ruta.

%~nx1 - expande **%1** solo a un nombre de archivo y extensión.

%~dp$PATH:1 - busca **%1** en los directorios listados en la variable de entorno **PATH** y se expande a la letra de unidad y ruta que se encontró primero.

%~ftza1 - expande %**1** a **DIR** como una línea de salida.

En el ejemplo anterior **%1** y **PATH** pueden ser reemplazados por otros valores válidos. Un número de argumento válido termina la sintaxis **%~.** Es posible que los modificadores %~ no se puedan usar con **%*.**

Ejemplo.

Supongamos que realizamos una programa o fichero principal.bat, que ejecutará dos procesos: procesoa.bat y procesob.bat. El contenido de principal.bat es el siguiente:

```
@ECHO OFF
CLS
CALL PROCESOA
CALL PROCESOB
PAUSE
CLS
```

El contenido de procesoa.bat es el siguiente:

```
@ECHO OFF
CLS
DIR c:\*.*
```

El contenido de procesob.bat es el siguiente:

```
@ECHO OFF
CLS
DATE
TIME
```